SYSTEM 86/330
OVERVIEW MANUAL

Order Number: 143898-001

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

REV.	REVISION HISTORY	PRINT DATE
-001	Original Issue.	1/82

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

The Information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

BXP	Insite	iSBC	Multibus
CREDIT	Intel	iSBX	Multimodule
i	int$_e$l	Library Manager	Plug-A-Bubble
ICE	Intelevision	MCS	PROMPT
iCS	Intellec	Megachassis	RMX/80
i^m	iOSP	Micromainframe	System 2000
iMMX	iRMX	Micromap	UPI

A635/582/ 2K DD

PREFACE

This manual provides overview information about the System 86/330 Microcomputer System, a high-performance system that combines standard Intel board products, state-of-the-art peripherals, the iRMX 86 Operating System, standard Intel language products, and debugging and diagnostic tools into a single package. The first five chapters of this manual provide general overview information which will be of particular interest to readers who are unfamiliar with Intel hardware and software products. The sixth chapter provides detailed information about the System 86/330 software; this information is intended for users who are familiar with the iRMX 86 Operating system but who have not used the System 86/330 Microcomputer System before.

PRODUCT SAFETY STANDARDS

The System 86/330 Microcomputer System is listed under UL standard 114 (Safety of Electronic Data Processing Units and Systems). It is also certified by the Canadian Standards Association, standard C22.2 154-1975 (Safety of Data Processing Equipment). The system complies with the International Electronics Commission standard IEC 435 (Safety of Data Processing Equipment). The system also conforms to the applicable RFI/EMI requirements of VDE 0871/6.78, VDE 0875/6.77 and FCC rule 47 CFR, part 15, subpart J (Emission Limits for Computing Devices).

However, the System 86/330 Microcomputer System generates, uses, and radiates radio frequency energy. Interference to radio communications (RFI) may result if the equipment is not installed and operated in accordance with the procedures in the SYSTEM 86/330 INSTALLATION AND MAINTENANCE MANUAL. The equipment was tested for compliance and complies with the limits for a Class A computing Device pursuant to Subpart J of Part 15 of FCC rules designed to provide reasonable protection against interference (RFI) for operation in a commercial environment. If the equipment is operated in a residential area, users are responsible, at their own expense, to take whatever measures are required to correct the likely occurrence of an unacceptable level of interference (RFI).

NOTATIONAL CONVENTIONS

This manual contains several examples of user dialog at a terminal. Throughout this manual, user input is underscored to distinguish it from system output.

RELATED PUBLICATIONS

The following manuals provide additional information that may be helpful to users of this manual. Chapter 5 contains a short description of each of the manuals.

Manual	Number
System 86/330 Installation and Maintenance Manual	143897
System 86/300 Series Diagnostic Reference Manual	143896
Introduction to the iRMX™ 86 Operating System	9803124
iRMX™ 86 Nucleus Reference Manual	9803122
iRMX™ 86 Terminal Handler Reference Manual	143324
iRMX™ 86 Debugger Reference Manual	143323
iRMX™ 86 Basic I/O System Reference Manual	9803123
iRMX™ 86 Extended I/O System Reference Manual	143308
iRMX™ 86 Loader Reference Manual	143318
iRMX™ 86 Human Interface Reference Manual	9803202
iRMX™ 86 Disk Verification Utility Reference Manual	144133
iRMX™ 86 System Programmer's Reference Manual	142721
iRMX™ 86 Programming Techniques Manual	142982
Guide to Writing Device Drivers for the iRMX™ 86 and iRMX™ 88 I/O Systems	142926
iRMX™ 86 Configuration Guide	9803126
iRMX™ 86 Installation Guide	9803125
EDIT Reference Manual	143587
Guide to Using iRMX™ 86 Languages	143907
8086/8087/8088 Macro Assembly Language Reference Manual for 8086-Based Development Systems	121627
8086/8087/8088 Macro Assembler Operating Instructions for 8086-Based Development Systems	121628

Manual	Number
PL/M-86 User's Guide for 8086-Based Development Systems	121636
iAPX 86, 88 Family Utilities User's Guide for 8086-Based Development Systems	121616
Run-Time Support Manual for iAPX 86, 88 Applications	121776
User's Guide for the iSBC™ 957B iAPX 86, 88 Interface and Execution Package	143979
iRMX™ 86 System Debug Monitor Reference Manual	143908
iSBC™ 016A/032A/064A/028A/056A RAM Memory Board Hardware Reference Manual	143572
iSBC™ 215 Winchester Disk Controller Hardware Reference Manual	121593
iSBX™ 218 Flexible Disk Controller Hardware Reference Manual	121583
iSBC™ 86/12A Single Board Computer Hardware Reference Manual	9803074
iSBC™ 337 Multimodule™ Numeric Data Processor Hardware Reference Manual	142887
iSBC™ 680/681 Multistore User System Package Hardware Reference Manual	162432

CONTENTS

PAGE

FIGURES

TABLES

CHAPTER 1. INTRODUCTION

The System 86/330 Microcomputer System is a comprehensive hardware and software package designed to solve sophisticated 16-bit OEM computational problems. It is available in two versions: a desk-top version shown in Figure 1-1 (System 86/330) and a rack-mountable version (System 86/331). When connected to a standard RS232-compatible terminal which you provide, the System 86/330 can be used to develop software for Intel's iRMX 86 board-based systems. When configured with your software products, it becomes a cost-effective execution system.

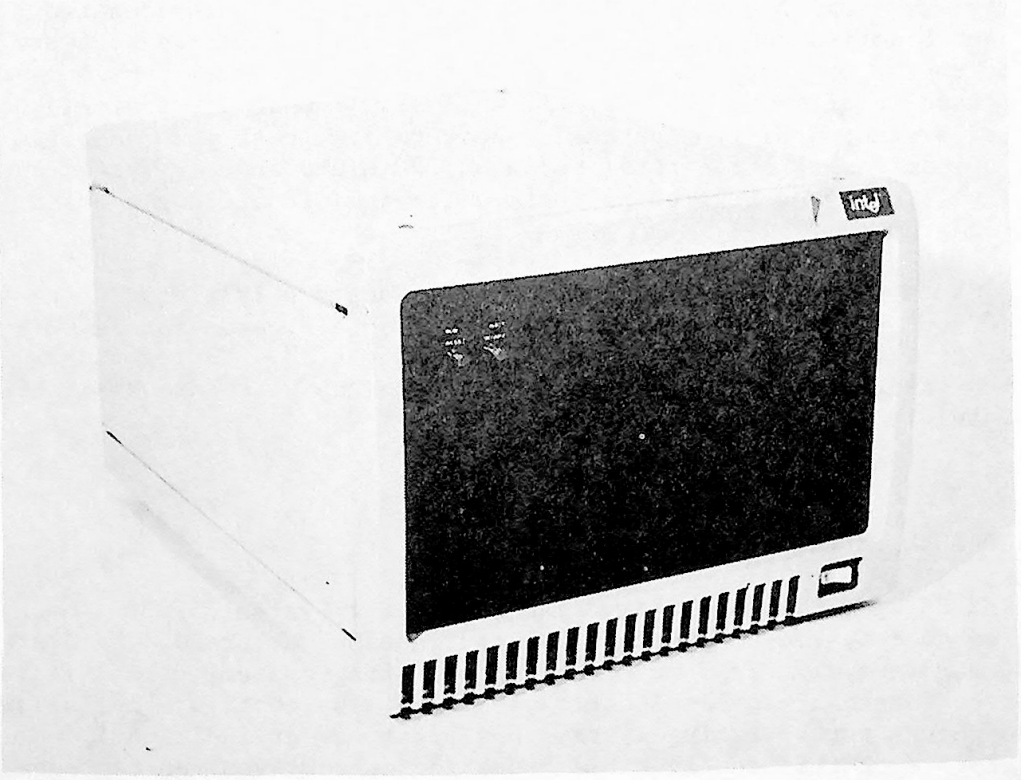

Figure 1-1. System 86/330 Microcomputer System

The System 86/330 has a number of features that are important to OEMs:

- Configurable hardware consisting of standared Intel boards and chassis
- Configurable, multitasking operating system designed for current and future Intel VLSI

- Support for multiprocessing and coprocessors
- Dual-purpose software environment (iRMX 86 program development and program execution)
- Available product service and technical support

This chapter discusses each of these features. It also discusses the contents of the remainder of this manual.

STANDARD INTEL HARDWARE

The System 86/330 Microcomputer System contains a standard Intel chassis, processor board, memory board, and disk controller board. All of these products are available individually from Intel. This provides two important benefits for you. First, because the application software you develop to run on your System 86/330 Microcomputer System runs on a system made up of standard parts, it will also run on Intel single board computer products that you purchase separately. Thus, you can build your first systems using the entire System 86/330 Microcomputer System and later build more-specialized products using individual single board computer products.

Second, because the system is made up of standard parts, you can easily upgrade your system when new, more-advanced Intel boards become available. You can also customize your System 86/330 Microcomputer System by adding other Multibus system bus boards which are available from Intel.

CONFIGURABLE, MULTITASKING OPERATING SYSTEM

The iRMX 86 Operating System available with the System 86/330 Microcomputer System complements the configurable nature of the System 86/330 hardware. It is a multitasking operating system, making it very effective for real-time applications. It is also configurable, allowing you to change the operating system to support new or additional hardware. With this configurable operating system, you can decrease your memory requirements by excluding parts of the operating system for which you have no use. You can also provide new or additional software features by extending the operating system.

SUPPORT FOR MULTIPROCESSING

The configurable nature of the System 86/330 hardware and software allows the System 86/330 Microcomputer System to support a multiprocessing environment. Because the hardware is configurable, you can add additional Intel processor boards to the system. Because the software is configurable, you can add Intel's Multibus Message Exchange software to

allow a processor board to communicate with other processor boards via the Multibus interface. This enables you to increase the general-purpose processing power of the System 86/330 Microcomputer System by adding by adding boards such as the iSBC 86/05 board and the iSBC 88/25 board. To add special-purpose processing power, you can also add intelligent, high-performance, microprocessor-based boards such as the iSBC 550 Ethernet communications controller (Ethernet is trademark of Xerox Corporation), the iSBC 544 serial communications controller, and the iSBC 88/40 analog measurement and control computer.

DUAL-PURPOSE SOFTWARE ENVIRONMENT

The System 86/330 Microcomputer System is a versatile and powerful vehicle for iRMX 86 program execution. However, unlike previous microcomputer systems, the System 86/330 Microcomputer System can also be used for iRMX 86 program development. The system contains a set of standard Intel language products which you can use to develop iRMX 86 application software and which you can include in your end product.

SERVICE AND SUPPORT

Intel offers several maintenance support options that you can tailor to your needs. These options include a board exchange program and on-site service (within areas serviced by Intel). Technical support is also available through qualified Field Application Engineers in Intel field offices. Contact your local Intel sales representative or distributor for more information.

ABOUT THE REST OF THIS MANUAL

The remaining chapters of this manual provide an overview of the software and hardware available with the System 86/330 Microcomputer System. These chapters are divided into two categories: general overview material intended for all readers and specific software information intended for readers who are already familiar with the iRMX 86 Human Interface.

Chapters 2 through 5 provide general overview material which you can read to get a feel for the kinds of hardware, software, and manuals which are available with the System 86/330 Microcomputer System.

- Chapter 2 provides an overview of the system hardware. It identifies each element of the system and points to other manuals where you can find more-detailed information.

- Chapter 3 provides an overview of the system software. It introduces you to the iRMX 86 Operating System and discusses each of the other software components. This chapter also refers you to other manuals for more-specific information.

- Chapter 4 contains a sample program development session. You can follow the instructions in this chapter to create a working PL/M-86 program.

- Chapter 5 contains an overview of the manuals available with the System 86/330 Microcomputer System. It lists the available manuals and provides a one-paragraph description of each manual.

Chapter 6 provides detailed software information about the System 86/330 Microcomputer System. In particular, it discusses the use of the preconfigured version of the iRMX 86 Operating System, a runnable version of the operating system that is already configured for the System 86/330 environment. This information is important for users who are familiar with the iRMX 86 Human Interface and who want to know how to communicate with the System 86/330 Microcomputer System via the Human Interface.

CHAPTER 2. HARDWARE OVERVIEW

The System 86/330 hardware consists of a chassis which contains a six-slot Multibus card cage, a switching power supply, a processor board, a disk controller board, a memory board, a 35M byte, 8-inch Winchester drive, and a 1M byte, 8-inch flexible disk drive. This chapter discusses each of these hardware elements.

CHASSIS

The chassis, Figure 2-1, is designed such that access for servicing the unit is both easy and convenient. The top cover is attached by four quarter-turn fasteners. The rear panel securing the power supply is attached with six screws. The front panel and both side panels are attached with ball-stud fasteners. The bottom cover is attached with four screws.

Mounted onto the front of the chassis is an illuminated power switch located at the lower right corner and a front panel with two pushbutton switches and two indicators.

The rear panel of the chassis supports I/O connectors whose functions are as follows:

Connector	Function
J1	RS232 serial I/O port for a CRT/Keyboard
J2	Parallel I/O communications port for an industry standard Centronics line printer interface
J3	Connector for additional Winchester drives
J4	Connector for additional flexible disk drives

Refer to the iSBC 680/681 USER SYSTEM PACKAGE HARDWARE REFERENCE MANUAL for additional information about the chassis.

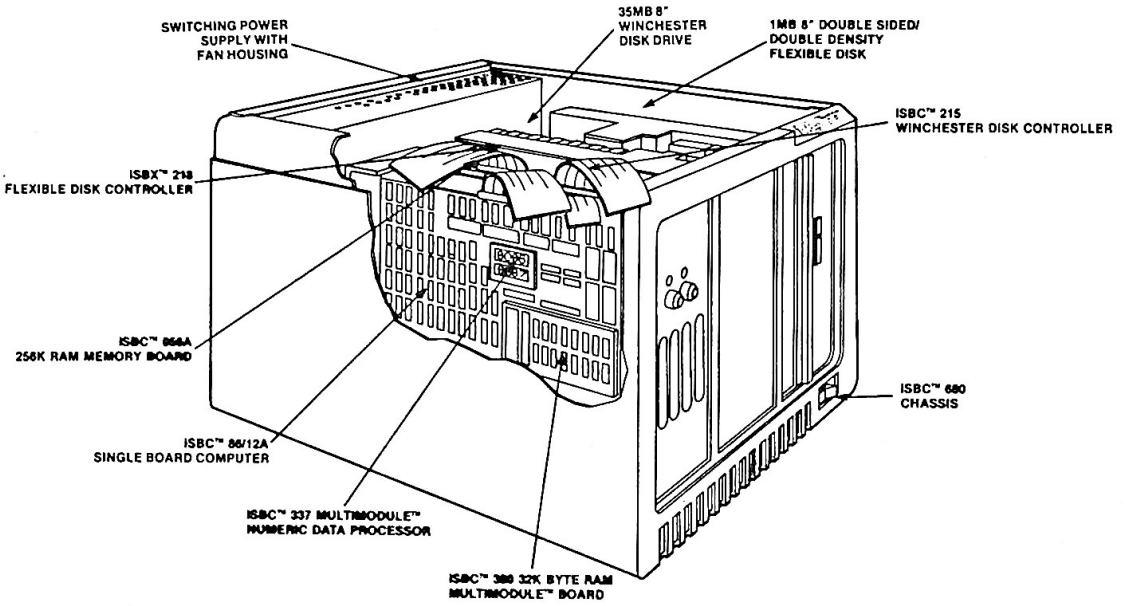

Figure 2-1. System 86/330 Internal Location Diagram

POWER SUPPLY

The power supply is mounted to the chassis at the rear panel. It is a switching power supply providing power to all elements of the system with sufficient reserve (92.5W) remaining for user-installable options. Refer to the SYSTEM 86/330 INSTALLATION AND MAINTENANCE MANUAL for the power supply specifications.

CARD CAGE

The six-slot card cage is mounted within the chassis on the extreme left side. The power supply routes DC power to all the slot connections of the card cage via connector P3. All boards are installed such that the component side faces to the right when viewed from the front. In the standard configuration, the boards fit into the slots of the card cage as follows (where slot connector J1 is on the far left side and slot connector J6 is on the far right side, when viewed from the front):

Slot Connector	Board
J1	Processor board
J2	Not available
J3	Available for expansion
J4	Available for expansion
J5	256K byte memory board
J6	Intelligent disk controller board

Hereafter, the descriptions within this manual assume that the system is in the standard hardware configuration.

The front panel is mounted in front of the card cage and interfaces to the backplane via connector P2. This card cage/backplane supports the Intel Multibus System Bus configuration for 8- or 16-bit data transfers and 20-bit addresses.

Bus contention between the processor board and the disk controller or other Multibus masters is resolved by a parallel resolution circuit within the backplane. The disk controller board has the highest priority and the processor board has the lowest priority.

PROCESSOR BOARD

The processor board consists of an Intel iSBC 86/12A Single Board Computer on which is mounted an iSBC 300A RAM expansion Multimodule board (with 32K bytes of dual-port RAM) and an iSBC 337 Numeric Data Processor Multimodule board.

The iSBC 86/12A board is a Multibus System Bus master with a 16-bit microprocessor (8086), 16K bytes of ROM memory, 32K bytes of dual port RAM memory, a Programmable Interval Timer (8253) that controls baud rates, a Programmable Interrupt Controller (8259A), a programmable USART (8251A) for RS232 serial I/O communications, and a Programmable Peripheral Interface (8255A) configured for an industry standard Centronics printer interface.

The iSBC 300A board is a RAM expansion Multimodule board extending the available on-board dual-port RAM to 64K bytes.

The iSBC 337 Numeric Data Processor adds extensive high-speed numeric processing capabilities to the iSBC 86/12A board. This Multimodule board includes an Intel 8087 numeric coprocessor. The coprocessor interface between the 8087 and the 8086 extends the instruction set by adding over 60 additional numeric instructions and six additional data types.

Refer to the iSBC 86/12A SINGLE BOARD COMPUTER HARDWARE REFERENCE MANUAL for a functional description of the iSBC 86/12A board and the iSBC 300A board. Refer to the iSBC 337 MULTIMODULE NUMERIC DATA PROCESSOR HARDWARE REFERENCE MANUAL for a functional description of the iSBC 337 board.

MEMORY BOARD

The RAM memory board is an iSBC 056A board providing the system with 256K bytes of additional memory.

Refer to the iSBC 016A/032A/064A/028A/056A RAM BOARD HARDWARE REFERENCE MANUAL for a functional description of the iSBC 056A board.

DISK CONTROLLERS

The disk controllers consist of an iSBC 215 Winchester disk controller to which is attached an iSBX 218 Flexible disk controller Multimodule board.

iSBC 215 WINCHESTER DISK CONTROLLER

The iSBC 215 Winchester disk controller is an 8089-based I/O processor which controls the Winchester drive in the Multibus System Bus environment. It provides full sector buffering, on-board ECC (error correction code), and automatic defective-track handling. It also features automatic error recovery and retry, and transparent data-error corrections. It can detect errors of up to 32 bits in length and can correct errors of up to 11 bits in length.

Refer to the iSBC 215 WINCHESTER DISK CONTROLLER HARDWARE REFERENCE MANUAL for a functional description of the iSBC 215 board.

iSBX 218 FLEXIBLE DISK CONTROLLER

The iSBX 218 Flexible disk controller is a double-wide Multimodule board which is attached to the iSBC 215 board. It controls all command, data, and status between the flexible disk drive and the processor board via the iSBC 215 board.

Refer to the iSBX 218 FLEXIBLE DISK CONTROLLER HARDWARE REFERENCE MANUAL for a functional description of the iSBX 218 board.

WINCHESTER DISK DRIVE

The Winchester disk drive is a 35M byte, non-removable media, 8-inch disk drive. The head/disk assembly is completely enclosed and does not need to be serviced. The Winchester drive is mounted in the peripheral bay with a filler panel mounted in front of the drive (see Figure 2-1).

FLEXIBLE DISK DRIVE

The flexible disk drive is a 1M byte double-sided, double-density drive. It is mounted in the right-most peripheral bay such that the door access button is positioned to the right of the diskette door as viewed from the front.

CHAPTER 3. SOFTWARE OVERVIEW

The following software is available with the System 86/330 Microcomputer System:

- A runnable version of the iRMX 86 Operating System (called the preconfigured version of the Operating System) which is already configured for use in a System 86/330 environment.
- A configurable version of the iRMX 86 Operating System, which you can extend to provide any additional support you need.
- A monitor and a debugger, which, when used together, provide a powerful debugging tool.
- System diagnostic programs, which check the System 86/330 hardware and help you find any possible hardware problems.
- A powerful, line-oriented text editor, for creating and modifying text files.
- Language translators and utilities, which allow you to assemble or compile your programs, link them together, place them in libraries, and assign absolute addresses to them.

This chapter discusses each of the software elements in general and refers you to other Intel manuals where you can obtain more information.

OPERATING SYSTEM SOFTWARE

The iRMX 86 Operating System is the heart of the System 86/330 software. It is a real-time, multiprogramming operating system which manages the resources of the System 86/330 Microcomputer System and allows it to respond to its environment. The next sections describe some of the features of the iRMX 86 Operating System.

PROCESSOR MANAGEMENT

The iRMX 86 Operating System contains the facilities to support the concurrent execution of multiple programs (called tasks). Although tasks can communicate with each other by invoking Operating System calls, each task runs independently of other tasks in the system. The Operating System resolves contentions between tasks by scheduling tasks for execution based on priority and readiness to run.

PRIORITY-BASED SCHEDULING

The Operating System provides a priority-based, event-driven scheduling mechanism that supports up to 256 different priority levels. The Operating System uses task priority to determine which task gains control of the processor; it always gives control of the processor to the highest-priority task that is ready to run. That task continues to run until a higher-priority interrupt occurs, the task requests resources that are not currently available, or the task makes a higher-priority task ready.

INTERRUPT-DRIVEN PROCESSING

The iRMX 86 Operating System is not an active monitor which runs constantly and wastes processor time. Instead, it is passive, using no processor resources until an interrupt occurs or a task makes a call to the Operating System. The interrupts represent requests for service from peripheral devices, such as the Winchester disk or an operator at the system terminal. Operating System calls represent requests for service from tasks. When the Operating System receives an interrupt or a system call, it stops the task that is currently running, responds to the request by performing some service, and resumes execution of the interrupted task.

In an iRMX 86-based application system, interrupt lines have corresponding iRMX 86 priorities. When an interrupt occurs, the Operating System calls a procedure to immediately handle the interrupt and, if necessary, it schedules a task of equivalent priority to perform further processing. By calling the first procedure, which is limited in the amount of processing it can do, the Operating System maximizes response time. By also scheduling an interrupt task whose priority corresponds to the priority of the interrupt, the Operating System allows you to perform extensive processing while at the same time guaranteeing that high-priority interrupts will be serviced before low-priority interrupts.

OBJECT-ORIENTED ARCHITECTURE

The iRMX 86 Operating System provides a number of data structures, called objects, which serve as the building blocks of your application system. It also provides a number of system calls which operate on these objects and manage the basic resources of the system. The seven basic objects are:

Tasks	Tasks perform the actual work of an iRMX 86 application system. Each task in the Operating System has the characteristics of a unique processor. Each has its own code, priority, stack, data area, registers, and status. The Operating System schedules tasks for execution based on their priority.

Segments	Segments are buffers of memory which the Operating System dynamically allocates to tasks when the tasks request memory. Segment size varies according to task needs.
Mailboxes	Mailboxes provide a mechanism for intertask communication. They are structures to which some tasks send objects and other tasks wait to receive objects. Mailboxes are generally used to pass segments from one task to another, but they can be used to transfer any kind of object.
Semaphores	Semaphores provide a mechanism for task synchronization and mutual exclusion. A semaphore is an integer counter which tasks can increment or decrement. This provides a means for one task to signal another that processing is complete or that resources are available.
Regions	Regions provide mutual exclusion by guarding a specific collection of shared data. A region permits only one task at a time to access the data guarded by the region.
Jobs	Jobs are the environments in which tasks exist. Each job contains a group of tasks, the objects used by the tasks, an object directory, and an area of memory. Jobs provides a multiprogramming environment, limiting the interraction between tasks in different jobs.
Composite objects	Composite objects are objects that a user can create to provide functions not available with the set of objects that already exists. The Operating System treats each new object the same as other objects, permitting the user to manipulate and obtain information about composite objects with the same calls that perform these functions on standard objects.

Figure 3-1 provides a pictorial description of a single job, showing how the job contains other iRMX 86 objects as well as a directory and an area of memory.

The Operating System provides a set of system calls that allow tasks to create objects, delete objects, manipulate objects, and request services from the Operating System. The iRMX 86 reference manuals describe these system calls in detail.

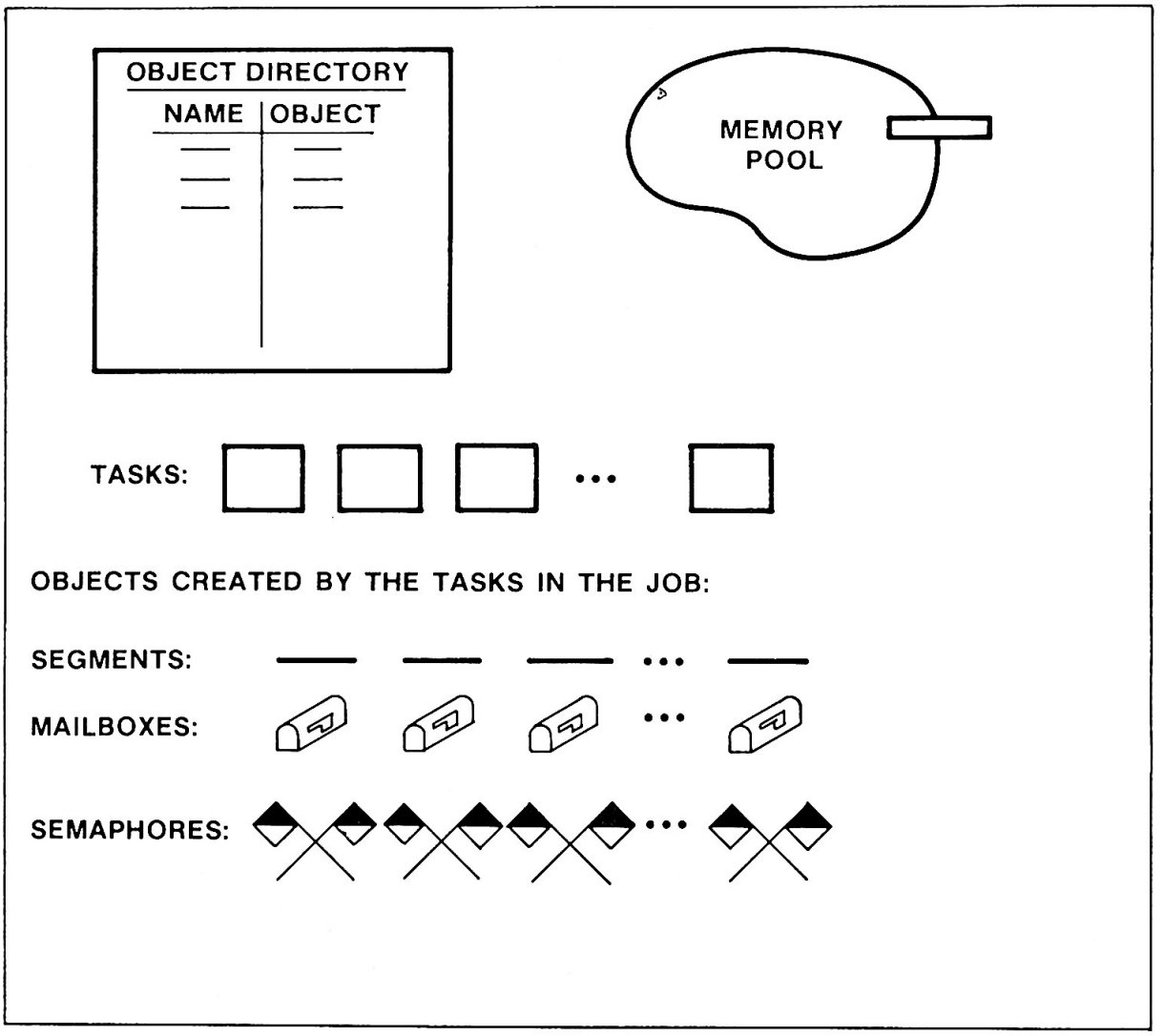

Figure 3-1. An iRMX™ 86 Job

ERROR PROCESSING

When a task issues an iRMX 86 system call, the results may not always be what the task expects. For example, a task might request memory that is not available, or it might use an invalid parameter. The Operating System provides a mechanism for transferring control, in the event of an error condition, to an error-handling routine. The Operating System permits hierarchical error handling and selective error processing.

Hierarchical error handling means that different levels of errors can be handled by different error-handling routines. For example, a system-wide error handler can process errors common to a number of tasks. Job-level error handlers can process application-specific errors. In addition, an individual task can have its own error handler, if necessary.

Selective error processing means that each task can select the type of errors to be processed by the error handlers. Errors are divided into two categories: programming errors (such as specifying invalid parameters) and environmental conditions (such as having insufficient memory to perform an operation).

DEVICE-INDEPENDENT I/O MANAGEMENT

The iRMX 86 Operating System provides a single set of system calls for managing I/O operations. These system calls allow an application program to be device independent, since a system call which performs an I/O operation on one device also performs the same operation on a different device. The iRMX 86 I/O System achieves this device independence by dividing its I/O processing into two parts: file drivers and device drivers.

File drivers are parts of the I/O System that communicate with files. There is a separate file driver for each type of file that the I/O System supports. The file types include:

Named files	Named files allow applications to refer to files by name. Each file name is cataloged in a directory to allow the file to be accessed with a minimum of overhead. Directories are special named files that store directory and access information about other named and directory files.
Physical files	Physical files provide a mechanism for applications to treat an entire I/O device as a single file. This type of file is typically used to communicate with simple devices such as printers and terminals.
Stream files	Stream files are mechanisms for communicating between tasks. They allow tasks to exchange data as if the data were written and then read from a first-in/first-out file.

The file drivers are completely device independent, allowing you to support many different kinds of devices without having to modify this part of the Operating System.

Device drivers are the device-dependent portions of the I/O System. There is a separate device driver for each type of I/O device in your system. The iRMX 86 Operating System supplies device drivers for many standard devices. It also gives you the capability to add your own device drivers to the I/O System in order to support your own special devices.

When your application program wishes to communicate with an I/O device, it first must invoke a system call that specifies a file driver and a device driver to be used with the device. This establishes the linkage between the file driver and the device driver, allowing your application program to communicate with a specific device. If, at a later time, you wish to switch to a different kind of device, you need change only one system call (the call that establishes the linkage between file and device driver). The remainder of your application program stays the same.

Refer to the iRMX 86 BASIC I/O SYSTEM REFERENCE MANUAL, the iRMX 86 EXTENDED I/O SYSTEM REFERENCE MANUAL, and the GUIDE TO WRITING DEVICE DRIVERS FOR THE iRMX 86 AND iRMX 88 I/O SYSTEMS for more information about file drivers, device drivers, and the system calls used to communicate with I/O devices.

HIERARCHICAL NAMING OF FILES

The iRMX 86 Operating System allows your application system to organize its named files into a tree-like structure like the one shown in Figure 3-2. There are two kinds of files in a hierarchical structure: data files (represented by triangles in the figure) and directory files (represented by rectangles in the figure). This hierarchy allows data to be grouped logically and accessed with a minimum of overhead.

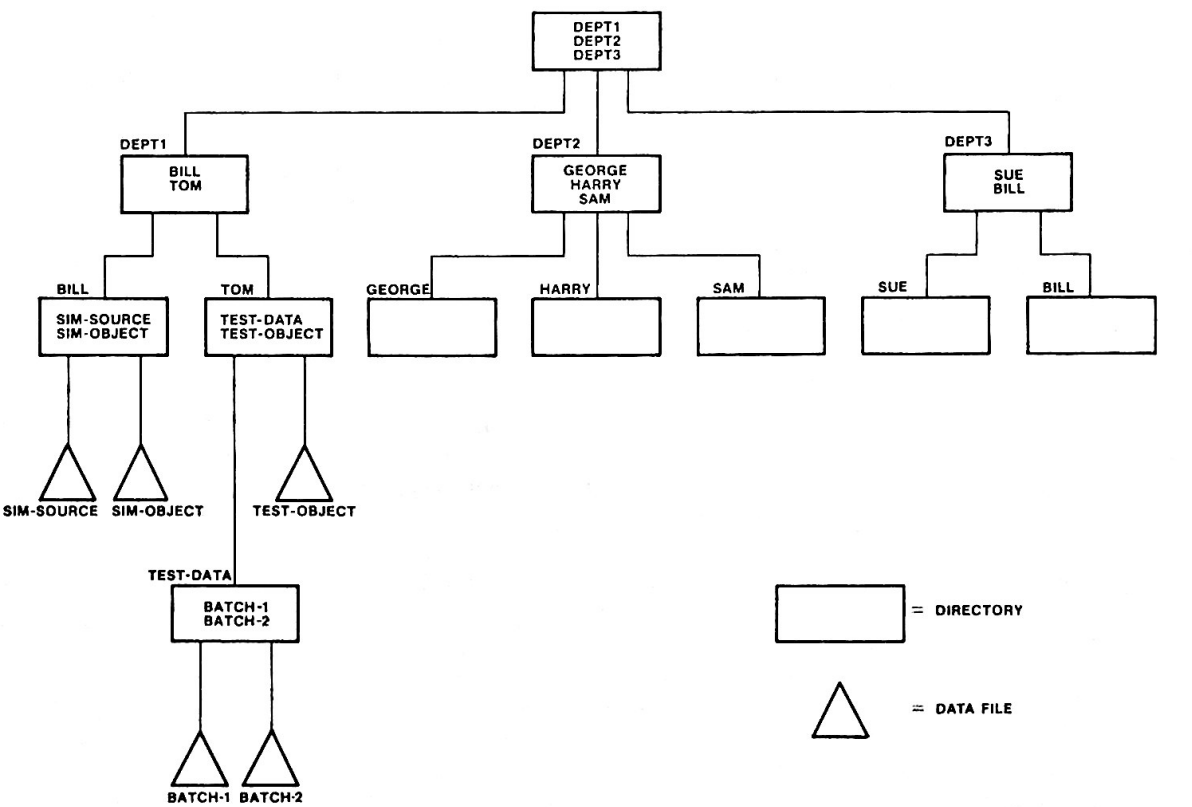

Figure 3-2. Hierarchical File Naming

CONFIGURABILITY

The iRMX 86 Operating System is the foundation on which a great variety of application systems can be built. As such, it is structured in a manner that allows users to select the particular features of the Operating System that they want and to discard any unused portions. This selection process is called configuration.

To aid in the configuration process, the Operating System is divided into a number of major pieces, called layers. When you configure the Operating System, you can select the layers you want for your application system and discard the rest. Figure 3-3 illustrates the layers that form the preconfigured version of the iRMX 86 Operating System. Other layers also exist.

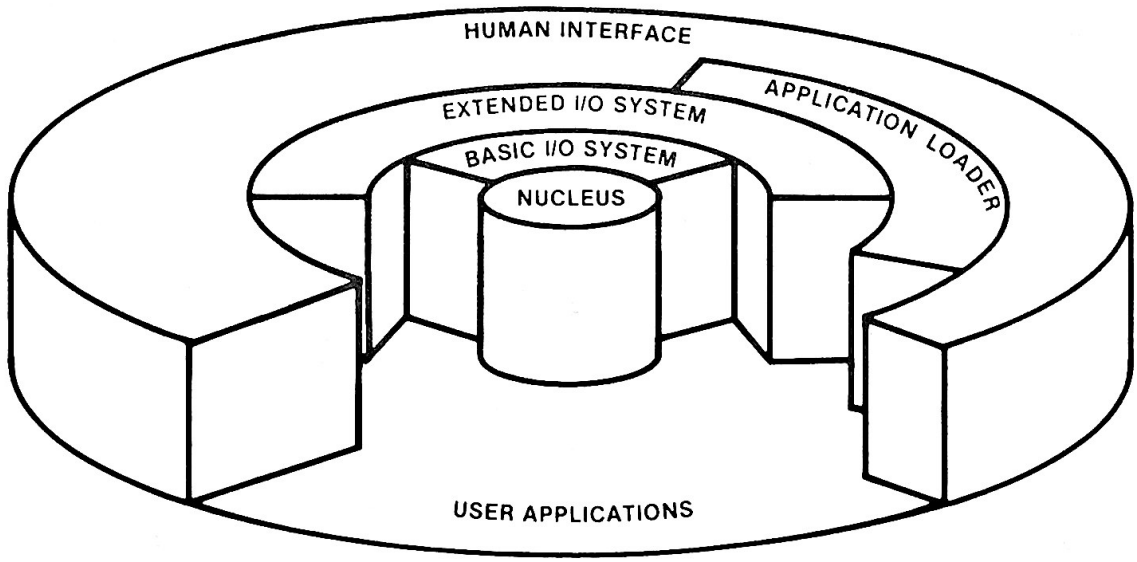

Figure 3-3. iRMX™ 86 Layers

The layers available with the configurable version of the iRMX 86 Operating System include:

Nucleus	The Nucleus is the core of the iRMX 86 Operating System and is required by every application system. It provides facilities that perform processor management and scheduling, interrupt management, memory management, object control, and error management. Refer to the iRMX 86 NUCLEUS REFERENCE MANUAL and the iRMX 86 SYSTEM PROGRAMMER'S REFERENCE MANUAL for detailed information about the Nucleus.
Terminal Handler	The Terminal Handler provides a real-time interface between a terminal and the application system. It is useful for users who require the ability to communicate with the system but who do not need the full services of an I/O System. Refer to the iRMX 86 TERMINAL HANDLER REFERENCE MANUAL for more information.
Debugger	The Debugger provides the facilities for interactively debugging tasks. It allows you to debug several tasks while the remainder of the system continues to run. Refer to the iRMX 86 DEBUGGER REFERENCE MANUAL for more information.
Basic I/O System	The Basic I/O System provides an extensive facility for device-independent I/O. It supplies all file drivers and a number of device drivers. It implements an asynchronous interface to I/O operations, allowing tasks explicitly to overlap I/O functions with other operations. Refer to the iRMX 86 BASIC I/O SYSTEM REFERENCE MANUAL and the iRMX 86 SYSTEM PROGRAMMER'S REFERENCE MANUAL for more information.
Extended I/O System	The Extended I/O System provides a higher-level interface to files than the Basic I/O System provides. The Extended I/O System provides a simple, synchronous interface to I/O operations, one which automatically performs read-ahead and write-behind buffering. This synchronous interface also allows tasks to use logical names to refer to files. Refer to the iRMX 86 EXTENDED I/O SYSTEM REFERENCE MANUAL and the iRMX 86 SYSTEM PROGRAMMER'S REFERENCE MANUAL for more information.
Application Loader	The Application Loader provides a simple mechanism for loading application code and data files from I/O devices into system memory. It can load absolute code into fixed locations, relocatable code into dynamically-allocated memory locations, and it can load files containing overlays.

Bootstrap Loader

The Bootstrap Loader provides a means of loading the entire application system into system memory from an I/O device. It can be configured to load from a specific device or to use the first device that becomes ready once the system has been started. It can also be configured to load a file specified by the operator at the terminal. A version of the Bootstrap Loader resides in PROM on the processor board, allowing you to immediately load your application system.

Human Interface

The Human Interface is the uppermost layer of the iRMX 86 Operating System. It is an interactive interface between the user and the application system. It gives the user the ability to invoke an application program from the terminal by specifying the name of the file that contains the program.

Supplied with the Human Interface are several programs called commands. The Human Interface commands perform many operations, including:

- Creating, copying, renaming, and deleting files
- Loading and starting application programs
- Formatting and verifying device volumes
- Backing up and restoring files on devices
- Reading commands from a file, instead of from the terminal
- Communicating with the iSBC 957B package to debug programs and to copy files to and from an Intel development system

The Human Interface also provides a number of system calls that the application program can invoke to access Human Interface services. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information.

In addition to being able to select the layers of the Operating System that you want to include in your application system, you can also configure the individual iRMX 86 layers. You can include or exclude features, include or exclude system calls, and specify hardware port addresses so that your customized Operating System can run on any iAPX 86- or iAPX 88-based system. Refer to the iRMX 86 CONFIGURATION GUIDE for more information about the configuration process.

Because the Operating System is configurable, it also allows you to add support for loosely-coupled multiprocessing. The Multibus Message Exchange (iMMX 800) software package provides the software needed to allow communications between processor boards via the Multibus interface. If you add the iMMX 800 package, your iRMX 86 tasks can communicate with tasks running on other processor boards. Refer to the iMMX 800 SOFTWARE REFERENCE MANUAL AND USER'S GUIDE (Order Number: 143808) for more information about iMMX 800 software.

The System 86/330 Microcomputer System includes both a configurable version of the iRMX 86 Operating System and a preconfigured version. With the preconfigured version, you can start running your System 86/330 Microcomputer System immediately upon power up. You can use this "standard" system to tailor the configurable version of the Operating System to your needs and generate your own customized application system. Then, you can bootstrap load your customized system and continue from there.

PRECONFIGURED OPERATING SYSTEM

One of the diskettes supplied with the System 86/330 Microcomputer System, the System 86/330 Installation Diskette, contains as one of its files a version of the iRMX 86 Operating System that you can run immediately. This version, called the preconfigured version of the Operating System, is configured to run on the standard System 86/330 hardware. As illustrated in Figure 3-3, it contains the following fully-configured layers:

- Nucleus
- Basic I/O System, complete with device drivers for the following devices:

  Terminal

  Flexible and Winchester disk drives (iSBC 215/218 controllers)

  Centronics-compatible line printer interface
- Extended I/O System
- Application Loader
- Human Interface

In addition to these layers, the preconfigured version also contains a job which initializes the iRMX 86 System Debug Monitor and then deletes itself. The System Debug Monitor is an extension of the iSBC 957B monitor which allows you to debug your application system by examining iRMX 86 data structures. These monitors are discussed later in this chapter.

The iRMX 86 Bootstrap Loader, which resides in PROM on the System 86/330 Microcomputer System, is used to load the Operating System from diskette and to begin system operation.

Chapter 6 contains further, detailed information about the preconfigured version of the Operating System.

MONITOR

In addition to the iRMX 86 Operating System, the System 86/330 Microcomputer System contains the iSBC 957B monitor, which resides in EPROM on the processor board. This monitor allows you to perform basic debugging and memory examination functions without the presence of the Operating System. With the monitor, you can perform the following operations:

- Set breakpoints in programs or single-step through code
- Examine and modify registers and memory
- Perform I/O via 8086 input and output ports
- Move and compare blocks of memory

Also, if you change the jumpers on the iSBC 86/12A board and use a parallel load cable to connect the parallel port of the iSBC 86/12A board to an Intel Microprocessor Development System, you can use the load features of the iSBC 957B package to transfer files between the development system and your System 86/330 Microcomputer System. Refer to the USER'S GUIDE FOR THE iSBC 957B iAPX 86, 88 INTERFACE AND EXECUTION PACKAGE for information about monitor commands and for information on how to set up your hardware to support the load features.

You can access the monitor in one of two ways:

- By pressing the INTRPT button. This method works if you include the System Debug Monitor (discussed in the next section) in your application system or if you connect the INTRPT button to the nonmaskable interrupt of iSBC 86/12A board. In both cases, pressing the INTRPT button interrupts the application system and gives control to the monitor, which waits for your entry.
- By entering the Human Interface DEBUG command. This loads a program into memory and gives control to the monitor, permitting you to examine the program in detail. The iRMX 86 HUMAN INTERFACE REFERENCE MANUAL describes the DEBUG command in more detail.

iRMX 86 SYSTEM DEBUG MONITOR

The iRMX 86 System Debug Monitor (SDB) is an extension of the iSBC 957B monitor. It adds additional commands to the monitor to allow you to examine iRMX 86 objects directly. If you include the SDB in your application system, you can use the SDB commands whenever you access the iSBC 957B monitor, after bootstrap loading the Operating System. Refer to the iRMX 86 SYSTEM DEBUG MONITOR REFERENCE MANUAL for more information.

The preconfigured version of the Operating System includes the SDB. The configurable version of the Operating System contains libraries for the SDB so that you can include the SDB in your customized Operating System.

SYSTEM DIAGNOSTIC PROGRAMS

Three diagnostic packages accompany the System 86/330 Microcomputer System. These packages include:

- System Confidence Test (SCT)
- System Diagnostic Test (SDT)
- System Analysis Test (SAT)

These diagnostic packages exercise all the System 86/330 hardware and check for errors. The following sections discuss the individual diagnostic packages.

SYSTEM CONFIDENCE TEST (SCT)

The SCT provides a basic check of each hardware element of the System 86/330 Microcomputer System. It resides in PROM on the processor board of the system (along with the monitor and the Bootstrap Loader) and runs when you apply power to the system or press the RESET button.

When the SCT starts running, it first performs a memory test and checks to see if it can communicate with the terminal. After approximately five seconds, it displays a series of asterisks at the terminal. When you enter an upper-case "U" at the terminal, it then generates a display that shows the progress of the remainder of the SCT. It updates this display as it completes each routine in the test. A period (.) indicates the successful completion of a portion of the test; a question mark (?) indicates an error condition. Figure 3-4 shows the display of a successfully-completed SCT. For more detailed information about the SCT, refer to the SYSTEM 86/300 SERIES DIAGNOSTIC REFERENCE MANUAL.

Upon successful completion, the SCT passes control to the iRMX 86 Bootstrap Loader which loads the Operating System from disk and starts it running.

```
SCT330, V1.0
  TEST                                        STATUS

USART/TIMER                                   GO
PIC          .*.                              GO
ROMCKSM      .                                GO
SPINNING UP >>>
PPI          ...                              GO
NDP          .                                GO
RAM TEST              TOTAL MEMORY = nnnnK
  ON BOARD                                    GO
  OFF BOARD                                   GO
  EXTENDED                                    GO

PARITY INT                                    GO
WINCHESTER                                    GO
FLOPPY                                        GO
CONTRLR                                       GO

SCT SUCCESSFUL...NOW BOOTING iRMX86
```

nnnnK = decimal number of K bytes found in the system

Figure 3-4. Successfully-Completed SCT Display

SYSTEM DIAGNOSTIC TEST (SDT)

The System Diagnostic Test (SDT) provides a means of isolating a hardware problem to a specific sub-assembly. It contains a number of subtests that you can specify to isolate problems to a particular board in the system.

SYSTEM ANALYSIS TEST (SAT)

The System Analysis Test (SAT) provides a means to exercise the system hardware by running the system software for extended periods of time. This permits isolation of subtle problems to a sub-assembly when those elusive intermittently-reported errors occur.

You invoke the SAT by entering a command at the terminal. The SAT then compiles a test program, links the program with necessary libraries and starts the program running as an iRMX 86 task. The test program checks out the various parts of the system and returns messages to the system console to indicate errors. Refer to the SYSTEM 86/300 SERIES DIAGNOSTIC REFERENCE MANUAL for detailed information about the SAT.

Unlike the SAT, the SDT does not run under the control of the iRMX 86 Operating System. To invoke the SDT, you must press the INTRPT button to access the monitor and then call the Bootstrap Loader to load the SDT in place of the Operating System. The SDT then starts running, allowing you to select various tests for execution. Refer to the SYSTEM 86/300 SERIES DIAGNOSTIC REFERENCE MANUAL for detailed information about the SDT.

TEXT EDITOR

The System 86/330 Microcomputer System contains a sophisticated, line-oriented editing facility called EDIT. You invoke EDIT by entering a command at the terminal. You can then use EDIT commands to create and modify files of text. EDIT provides a number of simple, one-character commands to:

- Add text (A)
- Insert text (I)
- Move test (M)
- Substitute text (S)
- Print text at the terminal (P)
- Read text from a file (R)
- Write text to a file (W)

These easy-to-use commands allow you, if you are unfamiliar with EDIT, to begin creating and modifying files of text immediately, without having to spend considerable time learning how to use the editor.

Later, when you become familiar with the basic editing facilities, you can take advantage of EDIT's more powerful features to perform extensive global editing functions with compact, one-line commands. You can also use the EDIT macro processor, which allows you to process a complex string of commands by entering a single command and which allows you to define blocks of text which can be included anywhere in the text file.

For more information about EDIT, refer to the EDIT REFERENCE MANUAL, which contains a tutorial to lead you step-by-step through the basic EDIT commands. It also contains a comprehensive description of each EDIT command. Also, the next chapter of this manual contains an example which shows you how to use EDIT when writing a program.

LANGUAGE TRANSLATORS AND UTILITIES

The System 86/330 Microcomputer System contains, as standard products, a number of language translators and utilities. Other optional language products are also available. The following sections describe these products.

STANDARD LANGUAGE PRODUCTS

The System 86/330 Microcomputer System contains several language translators and utilities which you can invoke by entering commands at the terminal. These translators and utilities allow you to compile or assemble programs, link programs together, assign absolute addresses to programs, create libraries of programs, and convert absolute object modules to hexadecimal format. The translators and utilities supplied with the System 86/330 Microcomputer System consist of:

ASM86	The 8086/8087/8088 macro assembler
PLM86	The PL/M-86 compiler
LINK86	The 8086 Linker, which combines individually-compiled object modules into a single, relocatable object module
LOC86	The 8086 Locater, which assigns absolute addresses to relocatable object modules
LIB86	The 8086 Librarian, which creates and maintains object module libraries
OH86	A program which converts absolute object modules to hexadecimal format

With these products you can create executable programs which you can invoke from the terminal. You can also create application tasks and configure the iRMX 86 Operating System to include these tasks. Finally, if you are an OEM (original equipment manufacturer) you can include these languages with your end product to give your customers the ability to program the System 86/330 Microcomputer System.

All the System 86/330 language products are compatible with those available on Intel Microcomputer Development Systems. Thus you can develop part of your application on a development system such as a Series III and part of it on the System 86/330 Microcomputer System.

For more information about the language translators and utilities, you should first refer to the GUIDE TO USING iRMX 86 LANGUAGES, which gives you some general information about invoking the language products in an iRMX 86 environment. Then you should refer to the individual manuals for the specific products. These manuals include:

8086/8087/8088 MACRO ASSEMBLY LANGUAGE REFERENCE MANUAL FOR 8086-BASED DEVELOPMENT SYSTEMS

8086/8087/8088 MACRO ASSEMBLER OPERATING INSTRUCTIONS FOR 8086-BASED DEVELOPMENT SYSTEMS

PL/M-86 USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS

iAPX 86, 88 FAMILY UTILITIES USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS

The next chapter also contains some examples of using the language translators and utilities.

OPTIONAL LANGUAGE PRODUCTS

In addition to the standard language products, you can also obtain the following Intel language products for use on the System 86/330 Microcomputer System:

Pascal-86	A Pascal compiler that provides a strict implementation of the proposed ISO standard. It also provides extensions of the language to allow you to write programs specifically for microcomputers.
FORTRAN-86	A FORTRAN compiler that provides total compatibility with existing FORTRAN-86 code, plus many new language features listed in the FORTRAN-77 language standard.

Contact your local Intel sales representative or distributer for more information about optional Intel language products.

Also, the iRMX 86 Operating System supports UDI (Universal Development Interface), a standard method for application programs to request operating system services. Because of this interface, a wide variety of optional language products, both compilers and interpreters, are available from independent software vendors for use on the iRMX 86 Operating System. Contact your local Intel sales representative or distributer for more information about the optional language products that are available through independent software vendors.

CHAPTER 4. USING THE SYSTEM

This chapter takes you step-by-step through a sample program development session on the System 86/330 Microcomputer System. If you are unfamiliar with the Human Interface commands or with the iRMX 86 language products, this chapter can help you get started using the system. It covers each step in the development process of a program, from starting up the system to running a program. If you wish, you can follow along on your own system, performing the same operations that this chapter lists. The operations include:

- Starting the system and bootstrap loading the Operating System
- Entering a PL/M-86 source program into a file by using the editor
- Compiling the program with the PL/M-86 compiler
- Using LINK86 to link the program with iRMX 86 interface libraries
- Running the program

The following sections of this chapter discuss the operations in more detail. The first of these sections lists some assumptions that this chapter makes about you System 86/330 hardware and software.

ASSUMPTIONS

This chapter makes the following assumptions about your System 86/330 hardware and software:

- A terminal configured for 9600 baud is connected to the J1 connector on the back of the System 86/330 Microcomputer System. This allows you to communicate with the system.
- The System 86/330 software is already installed on the Winchester disk. If you have not followed the installation procedures to transfer the software from the installation diskette to the Winchester disk, you should do so before reading further in this manual. The SYSTEM 86/330 INSTALLATION AND MAINTENANCE MANUAL describes this installation procedure.
- You are running the preconfigured version of the Operating System. If you have configured your own version of the Operating System, the system may respond differently than this chapter indicates.

STARTING THE SYSTEM

To start the system and bootstrap load the Operating System, perform the following steps:

1. Turn on the power to both the System 86/330 Microcomputer System and the terminal to which it is connected. The System 86/330 power switch is located on the lower right portion of the front panel. The System Confidence Test (SCT) starts running; in approximately five seconds it should begin displaing a series of asterisks (*) at the terminal.

2. Type an upper-case "U" at the terminal. This causes the SCT to continue.

3. The remainder of the SCT runs, displaying the information shown in Figure 3-4. The status column on the right of the display shows the status of the confidence tests. A "GO" indicates that the test passed; a "NO-GO" indicates that the test failed. The second test in the SCT, the PIC test, waits approximately five seconds for you to enter a character at the terminal. Enter an "I" character in response to the period (.) prompt. If you fail to enter a character when the SCT displays this prompt, a status of "NO-GO" appears for this test.

   When the SCT completes all tests successfully, it displays the following information:

   ```
   SCT SUCCESSFUL ... NOW BOOTING iRMX 86
   ```

   The Bootstrap Loader next loads the iRMX 86 Operating System into memory, a process that takes approximately one minute.

4. When the bootstrap load process completes, the Operating System displays the following information:

   ```
   SYSTEM 86/330  V1.0 : user = WORLD
   -DATE
   DATE:  01 JAN 78
   -TIME
   TIME:  00:00:00
   END SUBMIT LOGIN
   -
   ```

   The date and time messages inform you of the date and time that the system assumes. This is a reminder for you to set the correct date and time in the system. Do this by entering the DATE and TIME Human Interface commands, as follows (user input is underscored):

   ```
   -DATE dd mmm yy
   DATE: dd mmm yy
   -TIME hh:mm:ss
   TIME: hh:mm:ss
   -
   ```

where dd mmm yy indicates the current date (such as 01 MAR 82 or 23 APR 83) and hh:mm:ss indicates the current time (such as 11:45:00 or 14:30:45). Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for specific information on these commands. You should always set the data and time when you restart the system.

You can now proceed with your program development or enter any Human Interface commands you wish.

CREATING THE SOURCE CODE OF A PROGRAM

Now that you have bootstrap loaded the Operating System, you can use EDIT, the text editor, to enter the source code of a program into a file. To start this process, enter the following command:

```
-ED
```

EDIT responds by displaying a sign-on message and a prompt (the asterisk character), as follows:

```
iRMX 86 LINE EDITOR, Vx.x
*
```

The asterisk indicates that you are communicating with EDIT. You can now enter any of the commands described in the EDIT REFERENCE MANUAL.

To begin entering the program, use the add (A) command. Enter the following:

```
*A
```

You can now enter the text of a program. Figure 4-1 contains a sample PL/M-86 program that you can enter. This program displays the letters of the alphabet and asks you for permission to continue. It is a simple program that uses Human Interface and Extended I/O System calls to communicate with the terminal. You can examine the listing in Figure 4-1 to determine how the program works, or you can simply enter the program as it is listed.

If you have the iRMX 86 release diskettes available, you can eliminate several lines of code from this program. To do this, copy the files EIOS.EXT (from the Extended I/O System Release Diskette) and HI.EXT (from the Human Interface Release Diskette) and place them in your default directory. Then include in the program the lines:

```
$include(EIOS.EXT)
$include(HI.EXT)
```

in place of the section of the program that consists of the external declarations for the iRMX 86 system calls. These files (EIOS.EXT and HI.EXT) contain all the external declarations this program needs. The $include statements include the files in the compilation of the program.

```
alph:  DO;

DECLARE  TOKEN  LITERALLY  'WORD';

/*
 *  External declarations for iRMX 86 system calls
 */

rq$C$send$CO$response: PROCEDURE(response$p, response$max,
                                 message$p, excep$p)
                       EXTERNAL;
DECLARE
  response$p     POINTER,
  response$max   WORD,
  message$p      POINTER,
  excep$p   POINTER;
END rq$C$send$CO$response;

rq$exit$io$job: PROCEDURE(user$excep$code, return$data, excep$p)
                EXTERNAL;
DECLARE
  user$excep$code     WORD,
  return$data         POINTER,
  excep$p             POINTER;
END rq$exit$io$job;

/*
 *  End of external declarations
 */

/*
 *  Declare local variables
 */

DECLARE
  CR  LITERALLY '0Dh',
  LF  LITERALLY '0Ah',

  alphabet(26)  BYTE  DATA('ABCDEFGHIJKLMNOPQRSTUVWXYZ'),

  excep     WORD,
  n$letter  WORD,

  msg  STRUCTURE(
       length    BYTE,
       char      BYTE,
       CR$char   BYTE,
       LF$char   BYTE),
```

Figure 4-1. Sample Program

```
  read$buffer  STRUCTURE(
       length    BYTE,
       char(80)  BYTE),

  initial$msg(*)  BYTE  DATA (26, 'Letters of the alphabet:', CR, LF),
  continue$msg(*)  BYTE  DATA (21, 'Enter Y to continue: ');

/*
 *  Set up structures and write initial message
 */

n$letter = 0;
msg.length = 3;
msg.CR$char = CR;
msg.LF$char = LF;

call rq$C$send$CO$response(0, 0, @initial$msg, @excep);
read$buffer.char(0) = 'Y';

/*
 *  Display letters while response is Y
 */

DO WHILE  ((read$buffer.char(0) = 'Y')  OR
           (read$buffer.char(0) = 'y'));

  msg.char = alphabet(n$letter);
  call rq$C$send$CO$response(0, 0, @msg, @excep);
  call rq$C$send$CO$response(@read$buffer, SIZE(read$buffer),
                             @continue$msg, @excep);
  n$letter = (n$letter + 1) MOD 26;

END;

/*
 *  Exit
 */

call rq$exit$io$job(excep, 0, @excep);

END alph;
```

Figure 4-1. Sample Program (continued)

When you have finished entering the program (or if you wish to stop the process for some reason), enter a period (.) as the only character on a line. This informs EDIT that you are through entering text. This entry and response appear as:

```
.
*
```

Now you should check the program to ensure that you have entered it correctly. First, use the number (N) command to cause EDIT to display line numbers whenever it displays the text of the program. Enter the following:

```
*N
*
```

Next, use the print (P) command to display the first few lines of the program. Enter the following:

```
*1,10P
```

If you have entered the program exactly as shown in Figure 4-1, EDIT displays the first ten lines of the program, as follows:

```
1:      alph:  DO;
2:
3:      DECLARE  TOKEN  LITERALLY  'WORD';
4:
5:      /*
6:       *  External declarations for iRMX 86 system calls
7:       */
8:
9:      rq$C$send$CO$response: PROCEDURE(response$p, response$max,
10:                                       message$p, excep$p)
```

You should check these lines against Figure 4-1 to ensure that you have made no errors. If you have made errors, you can use EDIT commands to correct the text. For example, if you made a typing error in line 3, causing it to appear as:

```
3:      DECLARE  TOKEN  LITRALLY  'WORD';
```

you can use the substitute (S) command to correct the error. This command and response would appear as:

```
*3s/LITRALLY/LITERALLY/p
3:      DECLARE  TOKEN  LITERALLY 'WORD';
```

The "3" specifies the line number, the "s" indicates the substitute command, the values between the slashes indicate the old value and the new value, and the "p" causes EDIT to print the line after it makes the change. You can also use other commands to change the text; refer to the EDIT REFERENCE MANUAL for more information.

After you have examined the first ten lines, check the rest of the program for errors. You can enter a command such as:

```
*11,30p
```

to print out the next 20 lines, or you can use an over (O) command, such as:

```
*11o
```

to display the next "screen" of information (22 lines).

After verifying that you have entered the program correctly, use the write (W) command to write the program to a file. The following command writes the program to a file called ALPHA.P86 in the :PROG: directory:

```
*w :PROG:ALPHA.P86
:PROG:ALPHA.P86:  90 lines, 1850 bytes.
*
```

The response from EDIT that indicates the number of lines and bytes in the file may differ depending on how you enter the program.

EDIT does not save the program in a disk file unless you use the write command.

Finally, use the quit (Q) command to exit EDIT, as follows:

```
*q
-
```

The dash (-) prompt indicates that you have exited the editor and are again communicating with the Human Interface.

COMPILING THE PROGRAM

The next step in the program development process is compiling the program. To do that, enter the following command:

```
-PLM-86 :PROG:ALPHA.P86 COMPACT
```

This command invokes the PL/M-86 compiler, which compiles the program. The compiler automatically places the object code in the file :PROG:ALPHA.OBJ and the listing lines in the file :PROG:ALPHA.LST.

In response to the command, the compiler should display the following information:

```
iRMX 86 PL/M-86 COMPILER Vx.x
PL/M-86 COMPILATION COMPLETE.     0 WARNINGS,     0 ERRORS
-
```

However, if there is an error in the compilation, you should examine the listing file to determine the cause of the error and refer to the PL/M-86 USER'S GUIDE for more information. To display the listing file, enter the following command:

```
-COPY :PROG:ALPHA.LST
```

This command displays the entire listing file at the terminal. If the listing file is long and starts scrolling off the terminal screen, you can enter CTRL/s (press the CTRL key, and while holding it down, press the s key) to stop the listing. CTRL/q resumes the listing.

LINKING THE PROGRAM

Since the program invokes iRMX 86 system calls, you must link the program's object code with iRMX 86 interface libraries. These interface libraries satisfy external references generated by the calls in your program. You must link the program to the following interface libraries:

```
HPIFC.LIB       (Human Interface)
EPIFC.LIB       (Extended I/O System)
RPIFC.LIB       (Nucleus)
```

These interface libraries reside on the iRMX 86 release diskettes. Before you can link the program, you must copy these libraries from their corresponding release diskettes to the Winchester disk. For the purposes of this sample session, copy each file from its release diskette to a file of the same name in your default directory. For example, to copy HPIFC.LIB, place the Human Interface Release Diskette in the diskette drive and enter the following command:

```
-COPY :FD0:HPIFC.LIB TO HPIFC.LIB
```

Use similar commands to copy the remaining interface libraries from their release diskettes.

To link the program correctly, enter the following command:

```
-LINK86 :PROG:ALPHA.OBJ,    &
**      HPIFC.LIB,          &
**      EPIFC.LIB,          &
**      RPIFC.LIB           &
**    TO :PROG:ALPHA  BIND  SEGSIZE(STACK(800))
```

This command invokes LINK86, which links the program's object code with the three interface libraries and places the output module in a file called ALPHA under directory :PROG:. LINK86 automatically places the link map in the file :PROG:ALPHA.MP1. The BIND parameter is necessary for generating an output module which the Human Interface can load into any available dynamic memory. The SEGSIZE parameter is necessary for allocating a stack that is large enough to accommodate the iRMX 86 system calls.

In response to the command, LINK86 should display the following information:

```
iRMX 86 8086 LINKER Vx.x
-
```

If there is an error in the link process, LINK86 generates additional messages. In such a case, examine the link map (:PROG:ALPHA.MP1) for more specific information. Also refer to the iAPX 86, 88 FAMILY UTILITIES USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS for more information about LINK86.

RUNNING THE PROGRAM

After linking the program, you can run it by specifying the name of the file that contains the linked object code. In this case, the name of the file is :PROG:ALPHA. However, the directory :PROG: is special in that it is one of the directories that the Human Interface automatically searches whenever you enter a command and do not specify a directory name. (Refer to Chapter 6 for information about other such directories.) Therefore, to invoke the program, enter the following command:

```
-ALPHA
```

The program responds by displaying:

```
Letters of the alphabet:
A
Enter Y to continue:
```

If you enter a "Y", the program displays the next character; if you enter any other character, the program terminates.

CHAPTER 5. DOCUMENTATION

This chapter lists and briefly describes all of the documentation available with the System 86/330 Microcomputer System. This documentation includes system manuals, software manuals for all software elements of the system (iRMX 86 Operating System, language translators and utilities, diagnostics, monitor, and debug monitor), and hardware manuals for the chassis and for all boards. Additional copies of these manuals can be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

SYSTEM 86/330 MANUALS

The following manuals deal specifically with the System 86/330 Microcomputer System.

- SYSTEM 86/330 OVERVIEW MANUAL (Order Number: 143898)

  This is the manual you are currently reading. It is designed to introduce you to the System 86/330 Microcomputer System, give you an overview of the hardware and software elements of the system, and describe each of the Intel manuals available with the system. In addition, it describes some things you should know about the preconfigured version of the iRMX 86 Operating System.

- SYSTEM 86/330 INSTALLATION AND MAINTENANCE MANUAL (Order Number: 143897)

  This manual describes how to take the System 86/330 Microcomputer System as it is shipped from the factory, unpack it, set up the hardware, and install the preconfigured software. It also contains procedures to allow you to access and service the individual hardware elements.

- SYSTEM 86/300 SERIES DIAGNOSTIC REFERENCE MANUAL (Order Number: 143896)

  This manual describes the diagnostic programs available with the System 86/330 Microcomputer System. It discusses how to invoke the diagnostics and how to interpret the error messages that can result from running the diagnostics.

iRMX 86 MANUALS

The following manuals document the iRMX 86 Operating System.

- INTRODUCTION TO THE iRMX 86 OPERATING SYSTEM (Order Number: 9803124)

  This manual is designed to introduce engineers and managers to the iRMX 86 Operating System. It describes how the iRMX 86 Operating System can help you develop your application system in less time and at less expense.

- iRMX 86 NUCLEUS REFERENCE MANUAL (Order Number: 9803122)

  This manual documents the Nucleus, the central portion of the iRMX 86 Operating System required by all application systems. It provides overview information, discusses the functions of the Nucleus in detail, and contains detailed descriptions of the system calls available to application programmers.

- iRMX 86 TERMINAL HANDLER REFERENCE MANUAL (Order Number: 143324)

  This manual documents the Terminal Handler, a layer of the iRMX 86 Operating System that supports the use of a terminal as an I/O device for an application system. It describes the effects of entering certain special keyboard characters as well as how to use the Terminal Handler for supplying information to a task or receiving task output at a terminal. The preconfigured version of the iRMX 86 Operating System does not include the Terminal Handler. However, the special characters described in the manual are the same as those available with the System 86/330 Microcomputer System.

- iRMX 86 DEBUGGER REFERENCE MANUAL (Order Number: 143323)

  This manual documents the Debugger, a layer of the iRMX 86 Operating System that allows you to interactively examine your application system in order to find and correct errors. It contains introductory and overview material as well as detailed descriptions of all Debugger commands. The preconfigured version of the iRMX 86 Operating System does not include this Debugger. However, this manual is useful if you configure the Debugger into your application system.

- iRMX 86 BASIC I/O SYSTEM REFERENCE MANUAL (Order Number: 9803123)

  This manual describes the Basic I/O System, a layer of the iRMX 86 Operating System that provides flexible I/O features that are useful in a broad range of applications. It contains some introductory and overview material as well as detailed descriptions of the system calls available to application programmers.

- iRMX 86 EXTENDED I/O SYSTEM REFERENCE MANUAL (Order Number: 143308)

  This manual describes the Extended I/O System, a layer of the iRMX 86 Operating System that provides easy-to-use, more-automatic I/O features. It contains some introductory and overview material as well as detailed descriptions of the system calls available to application programmers.

- iRMX 86 LOADER REFERENCE MANUAL (Order Number: 143318)

  This manual describes the two loaders available with the iRMX 86 Operating System: the Bootstrap Loader and the Application Loader. It contains some introductory and overview material as well as detailed descriptions of the system calls available with the Application Loader.

- iRMX 86 HUMAN INTERFACE REFERENCE MANUAL (Order Number: 9803202)

  This manual documents the Human Interface, a layer of the iRMX 86 Operating System that provides an interactive interface between the user and the application system. It provides introductory and overview information, describes the commands available with the Human Interface, discusses the process of creating your own Human Interface commands, and contains detailed descriptions of the system calls available to the application programmer.

- iRMX 86 DISK VERIFICATION UTILITY REFERENCE MANUAL (Order Number: 144133)

  This manual documents the Disk Verification Utility, a software tool that runs as a Human Interface command, verifying and modifying iRMX 86 named and physical volumes. The manual describes how to invoke the utility, provides a detailed description of all utility commands, and discusses the format of iRMX 86 named volumes. You should read this manual if you suspect that one of your volumes is damaged or contains inconsistent information.

- iRMX 86 SYSTEM PROGRAMMER'S REFERENCE MANUAL (Order Number: 142721)

  This manual documents the more advanced features of the iRMX 86 Operating System, those normally reserved for system programmers. It includes discussions on using regions, extending the operating system by including new types, attaching I/O devices, and creating user objects. It also contains detailed descriptions of the system calls reserved for system programmers.

- iRMX 86 PROGRAMMING TECHNIQUES MANUAL (Order Number: 142982)

  This manual provides a number of programming techniques that can reduce the amount of time you spend designing and implementing your iRMX 86-based application system. It includes discussions on PL/M-86 size controls, interface procedures, INCLUDE files, timer routines, assembly language programming, job communication, configuration, deadlock, terminal I/O, and stack sizes.

- GUIDE TO WRITING DEVICE DRIVERS FOR THE iRMX 86 AND iRMX 88 I/O SYSTEMS (Order Number: 142926)

  This manual provides instructions that allow you to write your own iRMX 86 device drivers. You will need to read this manual if you want to communicate with devices for which the iRMX 86 Operating System does not already supply device drivers.

- iRMX 86 CONFIGURATION GUIDE (Order Number: 9803126)

  This manual provides the information that allows you to select the iRMX 86 layers that are appropriate for your application system, tailor them to meet your individual needs, and combine them with your own application software to form a functional application system. You do not need to read this manual if you plan to use only the preconfigured version of the Operating System.

- iRMX 86 INSTALLATION GUIDE (Order Number: 9803125)

  This manual provides information that allows you to set up your target system and install the iRMX 86 Operating System on that hardware. It also provides information about the iRMX 86 Patching Utility. If you plan to use the System 86/330 Microcomputer System as your target system, the portion of the manual that discusses the Patching Utility is the only portion that is relevant. However, if you plan to transfer your software to a board- or component-based hardware system, you may find the remainder of this manual useful.

LANGUAGE TRANSLATORS AND UTILITIES MANUALS

The following manuals document the language products that accompany the System 86/330 Microcomputer System.

- EDIT REFERENCE MANUAL (Order Number: 143587)

  This manual documents EDIT, an iRMX 86-based text editor which accompanies the System 86/330 Microcomputer System. It contains introductory and tutorial material as well as detailed descriptions of all EDIT commands.

- GUIDE TO USING iRMX 86 LANGUAGES (Order Number: 143907)

  This manual provides an overview of the language products that run in an iRMX 86 environment. It shows how to invoke the products from the Human Interface and lists the invocation controls for each product. It then refers you to other language and utilities manuals for detailed information about the products. You should read this manual before you read the other language and utilities manuals, because this manual provides additional information that you need to run the language products in an iRMX 86 environment. It also identifies portions of the other manuals that do not apply to the iRMX 86 versions of the language products.

- 8086/8087/8088 MACRO ASSEMBLY LANGUAGE REFERENCE MANUAL FOR 8086-BASED DEVELOPMENT SYSTEMS (Order Number: 121627)

  This manual documents the 8086/8087/8088 assembly language, ASM86. It describes the assembly language instructions as well as the data types, registers, codemacros, and the macro processing language.

- 8086/8087/8088 MACRO ASSEMBLER OPERATING INSTRUCTIONS FOR 8086-BASED DEVELOPMENT SYSTEMS (Order Number 121628)

  This manual describes how to invoke the assembler. It provides complete descriptions of all the assembler controls. It also describes how to link assembly language programs with PL/M-86 programs.

- PL/M-86 USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS (Order Number: 121636)

  This manual describes the PL/M-86 language and the use of the PL/M-86 compiler. It provides complete descriptions of all PL/M-86 language statements, discusses compiler invocation, and documents each of the compiler controls.

- iAPX 86,88 FAMILY UTILITIES USER'S GUIDE FOR 8086-BASED DEVELOPMENT SYSTEMS (Order Number: 121616)

  This manual contains detailed descriptions of the iAPX 86, 88 program development utilities available with the System 86/330 Microcomputer System. It describes LINK86, LOC86, LIB86, and OH86.

- RUN-TIME SUPPORT MANUAL FOR iAPX 86, 88 APPLICATIONS (Order Number: 121776)

  This manual describes the run-time aids that Intel offers for the iAPX 86, 88 family of processors. It discusses the run-time support available for iRMX 86 applications and for applications based on other operating systems. It also provides detailed information about logical record systems and the Universal Development Interface (UDI).

MONITOR AND DEBUG MONITOR MANUALS

The following manuals document the monitor and debug monitor that accompany the System 86/330 Microcomputer System.

- USER'S GUIDE FOR THE iSBC 957B iAPX 86, 88 INTERFACE AND EXECUTION PACKAGE (Order Number 143979)

  This manual provides general information, interfacing instructions, and programming information for the iSBC 957B loader and monitor. It provides detailed descriptions of the

loader and monitor commands and describes how to connect an Intel development system to an iSBC 86/12A board. It also contains configuration information, which may be of little importance to you since the monitor is already configured and available in PROM on your System 86/330 Microcomputer System.

- iRMX 86 SYSTEM DEBUG MONITOR REFERENCE MANUAL (Order Number: 143908)

  This manual documents the extension of the iSBC 957B monitor that allows you to examine iRMX 86 objects. This manual contains introductory and overview material as well as detailed descriptions of all debug monitor commands. It also describes configuration, installation, and invocation.

HARDWARE MANUALS

The following manuals document the individual hardware elements of the System 86/330 Microcomputer System.

- iSBC 016A/032A/064A/028A/056A RAM MEMORY BOARD HARDWARE REFERENCE MANUAL (Order Number: 143572)

  This manual provides information concerning the specifications, jumper configurations, programming considerations, and principles of operation of the iSBC 056A RAM memory board.

- iSBC 215 WINCHESTER DISK CONTROLLER HARDWARE REFERENCE MANUAL (Order Number: 121593)

  This manual provides information concerning the specifications, interfacing considerations, programming considerations, and principles of operation of the iSBC 215 Winchester Disk Controller board.

- iSBX 218 FLEXIBLE DISK CONTROLLER HARDWARE REFERENCE MANUAL (Order Number: 121583)

  This manual provides information concerning the specifications, jumper configurations, programming considerations, and principles of operation of the iSBX 218 Flexible Disk Controller board.

- iSBC 86/12A SINGLE BOARD COMPUTER HARDWARE REFERENCE MANUAL (Order Number: 9803074)

  This manual provides information concerning the specifications, jumper configurations, interfacing considerations, programming considerations, and principles of operation of the iSBC 86/12A Single Board Computer and the iSBC 300 RAM expansion board.

- iSBC 337 MULTIMODULE NUMERIC DATA PROCESSOR HARDWARE REFERENCE MANUAL (Order Number: 142887)

  This manual provides information concerning the specifications, jumper configurations, programming considerations, and principles of operation of the iSBC 337 Numeric Data Processor.

- iSBC 680/681 MULTISTORE USER SYSTEM PACKAGE HARDWARE REFERENCE MANUAL (Order Number: 162432)

  This manual provides information concerning specifications, installation considerations, and principles of operation of the iSBC 680 power supply and chassis.

CHAPTER 6. SOFTWARE CONFIGURATION

This chapter presents some detailed information about the preconfigured version of the iRMX 86 Operating System that accompanies the System 86/330 Microcomputer System. It provides this information for the following reasons:

- To inform you of the directory structure and logical names assumed by the preconfigured version of the Operating System so that you can use this information when entering Human Interface commands.
- To allow you to include the features of the preconfigured Operating System in any system that you configure.

The first portion of this chapter asssumes that you are familiar with the iRMX 86 hierarchical file structure and the Human Interface commands as described in the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL. The latter portion of this chapter assumes you are familiar with software configuration as described in the iRMX 86 CONFIGURATION GUIDE.

BOOTSTRAP LOADING

Whenever you press the RESET button on the System 86/330 Microcomputer System, the system runs the System Confidence Test (SCT) and, if the SCT completes sucessfully, bootstrap loads the iRMX 86 Operating System into memory. In order to do this, the iRMX 86 Bootstrap Loader, which performs this loading process, makes an assumption about which file to load. It assumes that the file containing the Operating System resides on the Winchester disk and has the pathname SYSTEM/RMX86. If such a file does not exist, the Bootstrap Loader cannot automatically load the Operating System.

There is also a mechanism that you can use to manually bootstrap load files into memory. You can use this mechanism under any of the following conditions:

- If the Winchester disk does not contain a file whose pathname is SYSTEM/RMX86
- If the SCT fails and you still want to bootstrap load the Operating System (when the SCT fails, it transfers control to the iSBC 957B monitor without bootstrap loading the Operating System)
- If you want to bootstrap load any file other than the preconfigured version of the Operating System (such as a customized version of the Operating System or a diagnostic program)

To perform this manual bootstrap loading process, you must press the INTRPT button to transfer control to the iSBC 957B monitor. Then you can use the B command to bootstrap load any file you specify. For example, suppose you have configured your own version of the Operating System and placed it in the file USER/PROG/MYSYS on the flexible disk drive. You can bootstrap load this file into the System 86/330 Microcomputer System by pressing the INTRPT button and entering the following command:

```
.B :WF0:/USER/PROG/MYSYS
```

Notice that the format for specifying file names is slightly different than when using the Human Interface. The iSBC 957B device names are different and include:

:WF0:	Flexible disk drive
:W0:	Winchester disk drive

You must also place a slash (/) between the device name and the beginning of the file name.

The Bootstrap Loader is configured with :W0: as the default device and SYSTEM/RMX86 as the default file. Therefore, to bootstrap load the Operating System from the default file on the Winchester disk, you could enter the command:

```
.B
```

This command would load the file SYSTEM/RMX86 from the Winchester disk. To load a file other than SYSTEM/RMX86 from the Winchester disk (for example, the file USER/PROG/SYS2), you could enter the command:

```
.B /USER/PROG/SYS2
```

The device name is not required, but the initial slash is required. Refer to the iRMX 86 LOADER REFERENCE MANUAL for additional information on how to enter file names when communicating with the Bootstrap Loader.

The System 86/330 Installation Diskette, which accompanies the system, contains two Operating System files which can be loaded by the Bootstrap Loader. These files have the pathnames SYSTEM/RMX86 and SYSTEM/RMX86.WD0.

The first file, SYSTEM/RMX86, contains a preconfigured version of the Operating System which assumes that a number of required directories and data files reside on the flexible disk. By interrupting into the monitor and bootstrap loading this file, you can bring up the Operating System before formatting the Winchester disk. You can then format the Winchester disk and install the files and directories onto it. A SUBMIT file on the installation diskette, USER/INSTALL.CSD performs this operation. Refer to the SYSTEM 86/330 INSTALLATION AND MAINTENANCE MANUAL for more information on how to use this file.

The second file, SYSTEM/RMX86.WD0, is a preconfigured version which assumes that the required directories and data files reside on the Winchester disk. The SUBMIT file USER/INSTALL.CSD copies this file to

the Winchester disk and gives it the pathname SYSTEM/RMX86. Later, when you press the RESET button, the Bootstrap Loader automatically loads the Operating System from this file.

Because the bootstrap load files are slightly different, you should remember that when you bootstrap load the system from the Winchester disk, the Operating System assumes that the Winchester disk is the system disk. And, when you bootstrap load the system from the Installation Diskette, the Operating System assumes that the flexible disk is the system disk.

DIRECTORY STRUCTURE

The preconfigured version of the iRMX 86 Operating System assumes that one of your secondary storage devices (the flexible disk or the Winchester disk) is the system disk (refer to the previous section for more information). When the Operating System starts running, it searches the system disk for certain directories and files. If these directories and files do not exist, the Operating System might not initialize correctly. The installation diskette, supplied as part of the System 86/330 release materials, is set up with the correct directory structure. You should ensure that the Winchester disk also contains this directory structure so that you can use it as you system disk. (The SUBMIT file USER/INSTALL.CSD sets up the Winchester disk with the correct directories and copies a number of data files under these directories. (Refer to the SYSTEM 86/330 INSTALLATION AND MAINTENANCE MANUAL for more information.) Figure 6-1 illustrates the directory structure assumed by the preconfigured version of the Operating System.

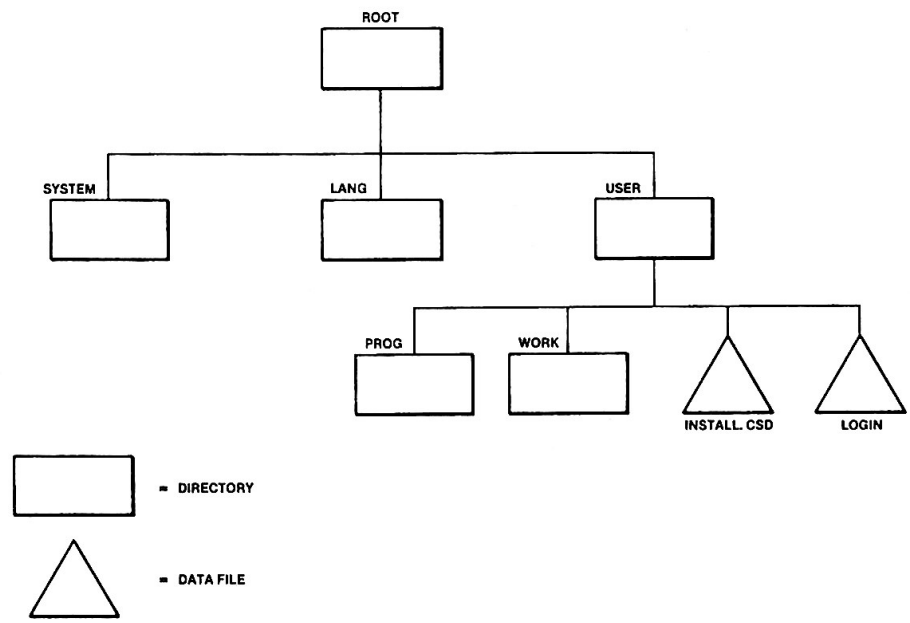

Figure 6-1. System 86/330 Directory Structure

As you can see from Figure 6-1, the Operating System assumes that your system disk contains five directories (other than the root directory of the volume, which is always present). On the installation diskette, these directories serve the following purposes:

SYSTEM — This directory contains all the Human Interface commands, the iRMX 86 interface libraries, and the bootstrap load file (named RMX86).

LANG — This directory contains the editor, the iRMX 86 language translators, and the language utilities.

USER — This is your default directory. Unless you specify otherwise, all directories and files that you create are listed in this directory. When you enter the DIR command without parameters, the Operating System displays the entries in the USER directory.

PROG — This is a directory under the USER directory which you can use to store your programs.

WORK — This is a directory under the USER directory which the language translators and utilities use to store temporary and work files.

The directory USER contains two additional files, data files called INSTALL.CSD and LOGIN. INSTALL.CSD is a SUBMIT file which formats the Winchester disk and copies all the directories and data files from the installation diskette to the Winchester disk. You should submit INSTALL.CSD initially to install your software on the Winchester disk. Refer to the SYSTEM 86/330 INSTALLATION AND MAINTANANCE MANUAL for more information.

LOGIN is also a SUBMIT file. However, it is different from other SUBMIT files in that whenever you bootstrap load the Operating System, the Operating System automatically submits the file LOGIN. Refer to the "LOGIN File" section of this chapter for more information.

You can create as many additional directories and data files as you wish. However, the preconfigured version of the Operating System requires that the directories SYSTEM, LANG, USER, PROG, and WORK be present. If you reconfigure the Operating System, you can change this requirement. Refer to the iRMX 86 CONFIGURATION GUIDE for more information.

LOGICAL NAMES

During initialization, the iRMX 86 Operating System obtains connections to certain devices and files and catalogs these connections as logical names. This makes using the Human Interface easier for the following reasons:

- The Winchester and flexible disk devices are already attached. Thus, when you refer to these devices, you can use their logical names immediately, instead of having to enter the ATTACHDEVICE command to attach the devices.
- Certain directories have logical names so that you can refer to them without having to enter their complete pathnames.

The following sections discuss in more detail the logical names that the Operating System assigns. The sections divide the discussion into device logical names and directory logical names.

DEVICE LOGICAL NAMES

Table 6-1 lists the logical names for the devices that the preconfigured version of the Operating System assigns during its initialization. It also lists their physical names and their characteristics.

You can use the logical names listed in Table 6-1 when you enter Human Interface commands. For example, if you want to display the root directory of a volume in the flexible disk drive, you can enter the following command:

```
-DIR :FD0:
```

The Operating System then lists files in the directory.

Table 6-1. Logical Names of Devices

LOGICAL NAME	DEVICE	PHYSICAL NAME	CHARACTERISTICS
:WD0:	Winchester disk	IW0	1024-byte device granularity
:FD0:	Flexible disk	WFDD0	double-sided, double-density, 256-byte device granularity
:LP:	Line printer	LP	Centronics line printer interface; do not use unless printer is connected
:CI:	Console input	T0	
:CO:	Console output	T0	

As you can see from Table 6-1, the Operating System establishes a logical name for a line printer, even if you do not have a line printer connected to your system. However, you must not use :LP: as a logical name in any Human Interface command unless you actually have a line printer connected to the system. Any attempt to do so causes the system to hang, requiring you to reinitialize the system by pressing the RESET button.

The physical names listed in Table 6-1 are actually the names of blocks of information called device unit information blocks (or DUIBs) which are set up during the configuration of the Basic I/O System. These DUIBs describe the devices to the Basic I/O System, so that the Basic I/O System knows how to communicate with the devices. The preconfigured version of the Operating System contains several DUIBs, some of which describe the same device but in slightly different manners. (For example, one DUIB for the flexible disk drive may assume that the drive contains a double-sided, double-density diskette and another DUIB for the same device may assume that the drive contains a single-sided, single-density diskette.) During initialization, the Operating System attaches the devices based on information supplied to it in configuration parameters. A parameter for each device specifies the physical name of the device (DUIB name). This tells the Operating System which one of several possible DUIBs to use.

After system initialization, you can change the characteristics of the flexible disk drive, assign different logical names, or attach new devices to the system. You can change the characteristics of the flexible disk drive or assign a different logical name to any device by first using the DETACHDEVICE command to detach the device. Then you can use the ATTACHDEVICE command to reattach the device with either a different physical name or a different logical name. This allows you to

change the characteristics of the device or change its logical name without having to reconfigure the entire system. For example, suppose you wanted to use a single-density diskette with 128-byte granularity in your flexible disk drive. To do this you would have to enter the following commands:

```
-DETACHDEVICE :FD0:
:FD0:, detached
-ATTACHDEVICE WF0 AS :FD0:
WF0 attached as :FD0:
-
```

These two commands tell the Operating System first to break the connection to the device and then to reestablish the connection assuming the characteristics listed in the DUIB named WF0. WF0 is a physical name for a single-density drive with 128-byte granularity.

To attach a new device to the system, you would enter just the ATTACHDEVICE command. For example, if you connected an additional flexible disk drive to your system, you could attach it by entering the following command:

```
-ATTACHDEVICE WFD1 AS :FD1:
WFD1 attached as :FD1:
-
```

Table 6-2 lists the physical names available for use in the preconfigured version of the Operating System. This table also lists the characteristics associated with these physical names. Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for more information about ATTACHDEVICE and DETACHDEVICE.

Table 6-2. Physical Device Names

PHYSICAL NAME	DEVICE	CHARACTERISTICS
WF0	Flexible disk	Single-sided, single-density, 128-byte granularity, unit 0
WF1	Flexible disk	Single-sided, single-density, 128-byte granularity, unit 1
WFD0	Flexible disk	Single-sided, double-density, 256-byte granularity, unit 0
WFD1	Flexible disk	Single-sided, double-density, 256-byte granularity, unit 1
WFDD0	Flexible disk	Double-sided, double-density, 256-byte granularity, unit 0
WFDD1	Flexible disk	Double-sided, double-density, 256-byte granularity, unit 1
IW0	Winchester disk	1024-byte granularity, unit 0
IW1	Winchester disk	1024-byte granularity, unit 1
T0	Terminal	
LP	Line printer	
STREAM	Stream file device	
BB	Byte bucket	

If you configure your own version of the Operating System, you can add DUIBs, thereby allowing you to support different devices, different characteristics the the same devices, and additional units. Refer to the iRMX 86 CONFIGURATION GUIDE for more information.

DIRECTORY LOGICAL NAMES

In addition to assigning logical names to devices, the Operating System assigns logical names to several directories. This allows you to refer to the directories by their logical names, instead of having to specify their entire pathnames. Table 6-3 lists the logical names that the preconfigured version of the Operating System assigns. It also lists the pathnames that correspond to these logical names. All pathnames listed in Table 6-3 are assumed to reside on the system disk.

Table 6-3. Directory Logical Names

LOGICAL NAME	CORRESPONDING PATHNAME	DESCRIPTION
:SYSTEM:	SYSTEM	Directory containing Human Interface commands, iRMX 86 interface libraries, and bootstrap load file.
:$:	USER	Default directory that the Operating System assumes when you omit the beginning portion of a pathname.
:PROG:	USER/PROG	Directory which you can use to store your own programs.
:WORK:	USER/WORK	Directory which the language translators and utilities use to store temporary and work files.

ACCESSING DIRECTORIES

Once you initialize the Operating System, you can enter Human Interface commands to manipulate and examine data files and directories. In order to do this effectively, you should be aware of your default directory and how to access other directories.

The preconfigured version of the Operating System assigns the directory USER as your default directory. Refer to Figure 6-1 to see where this directory fits in the overall directory structure of the system disk. When you enter a file name at the system console and do not explicitly specify a directory with that file, the Operating System assumes that the file resides in directory USER. For example, if you enter the command:

```
DIR
```

the Operating System displays a listing of the directory USER. You could also obtain the same results by entering the following command:

```
DIR :$:
```

The logical name :$: also refers to your default directory.

However, if you want a listing of one of the directories on the same hierarchical level as USER, such as SYSTEM or LANG (refer to Figure 6-1), you cannot simply enter a command such as:

```
DIR SYSTEM          (incorrect entry)
```

This command would attempt to list a directory called SYSTEM that resides in your default directory (that is, a directory whose pathname is USER/SYSTEM). Instead, to obtain a listing of the directory SYSTEM, you should enter one of the following commands:

```
DIR :SYSTEM:
DIR /SYSTEM
DIR ^SYSTEM
```

The first command in the previous list, DIR :SYSTEM:, lists the directory corresponding to the logical name :SYSTEM:. Since the logical name :SYSTEM: corresponds to the directory SYSTEM, this command obtains the desired results.

In the second command, DIR /SYSTEM, the slash (/) preceding the SYSTEM means that the Operating System, when searching for the directory, ignores your default directory and assumes that SYSTEM resides in the root directory of the system disk. This also obtains the desired results. In general, whenever you begin a file name with a slash (/), the Operating System assumes that the path component that follows the slash resides in the root directory of your system disk. Each additional slash in the pathname causes the Operating System to search one level down in the file tree for the next path component.

In the third command, DIR ^SYSTEM, the up-arrow (↑) or circumflex (^) preceding the SYSTEM means that the Operating System, when searching the directory, begins at your default directory, goes one level up in the file tree, and then searches for the directory SYSTEM. This means that the Operating System searches for SYSTEM in the root directory, where it actually resides. In general, whenever you use an up-arrow or circumflex instead of a slash to separate path components, the Operating System searches one level up instead of one level down for the next path component.

DIRECTORY SEARCH

When you invoke any program from the Human Interface, whether it be a Human Interface command, a language product such as a compiler, or a program that you have written, you invoke the program by entering the name of the file that contains the program. If you specify the complete pathname of the program, the Operating System obtains the program from the device and directory that you specify and loads and executes the program. If the pathname you specify does not exist, the Operating System returns an error message. However, if you specify just the last component of the program's pathname, the Operating System searches several directories for the program. It does not return an error message until it has searched each of the directories. The directories that the preconfigured version of the Operating System searches, in order, are as follows:

:$:	default directory
:PROG:	program directory
:SYSTEM:	system directory
/LANG	languages directory

Therefore, if you create a program called myprog in the :PROG: directory, you can invoke that program by entering one of the following:

```
myprog
:PROG:myprog
:WD0:USER/PROG/myprog
```

The last two methods of invoking the program are unnecessarily long, because the Operating System automatically searches the :PROG: directory for the program. In the same manner, you can invoke all the Human Interface commands and the iRMX 86 language products without specifying their complete pathnames; they also reside in directories that the Operating System automatically searches.

You can take advantage of the fact that the Operating System searches directories in a particular order. For example, suppose you write your own copy command, one that provides more or different functions than the Human Interface COPY command. If you want to invoke your own program whenever you type the command "COPY", you can simply place your copy program in a file called COPY in the :PROG: directory. Then, when you enter the command "COPY", the Operating System will invoke your copy program instead of the Human Interface COPY command. It does this because it searches the :PROG: directory before it searches the :SYSTEM: directory; thus it finds your program first and executes that program.

If you still want to be able to invoke the Human Interface COPY command, you can do so by entering the complete pathname, that is, by entering one of the following:

```
:WD0:SYSTEM/copy
:SYSTEM:copy
```

LOGIN FILE

Whenever the preconfigured version of the Operating System starts running, it automatically invokes a SUBMIT file called LOGIN, which resides in the directory USER. As released, this SUBMIT file contains the following commands:

```
DATE
TIME
```

When you bootstrap load the Operating System, it displays a sign-on message and then runs the SUBMIT file LOGIN. The following lines are displayed at the system console:

```
SYSTEM 86/330 Vx.x : user = WORLD

-DATE
DATE: 01 JAN 78
-TIME
TIME: 00:00:00
-END SUBMIT LOGIN
```

This SUBMIT file reminds you to use the DATE and TIME Human Interface commands to set the system date and time. Otherwise, the Operating System assumes the system date and time to be the values listed in the display. You should always set the date and time when restarting the system to permit the system to record accurate creation and access dates for any files you create or access. The creation and access dates can be important if you use the BACKUP and RESTORE Human Interface commands. (Refer to the iRMX 86 HUMAN INTERFACE REFERENCE MANUAL for information about the BACKUP and RESTORE commands.) The System Analysis Test (SAT) also requires accurate values for the date and time.

By using the editor, you can modify the LOGIN file to contain other commands you want the Operating System to invoke. Then whenever you bootstrap load the Operating System, the commands in this file will be invoked.

CONFIGURATION FILES

A complete set of iRMX 86 release diskettes is available with the System 86/330 Microcomputer System. Using the files on these diskettes, you can create a customized version of the iRMX 86 Operating System that runs on the System 86/330 Microcomputer System or other iAPX 86, 88-based products. Although the iRMX 86 CONFIGURATION GUIDE is the primary source of configuration information, the following sections also discuss iRMX 86 configuration and the configuration files. These sections cover the following topics:

- Copying the release diskettes to the Winchester disk
- Modifying the iRMX 86 configuration and SUBMIT files so that they are compatible with System 86/330 device names

- Modifying the iRMX 86 configuration files to produce the features available with the preconfigured version of the Operating System

COPYING DISKETTES TO THE WINCHESTER DISK

The configurable version of the iRMX 86 Operating System resides on several diskettes. These diskettes include:

- Nucleus Release Diskette
- Terminal Handler and Debugger Release Diskette
- Basic I/O System Release Diskette
- Extended I/O System Release Diskette
- Application Loader and Bootstrap Loader Release Diskette
- Human Interface Release Diskette

There are several other diskettes in the iRMX 86 release package, but they may not be very useful to System 86/330 users.

To make the process of configuration on the System 86/330 Microcomputer System easier, it is advisable to copy all the files on the iRMX 86 release diskettes to the Winchester disk before starting the configuration process. Since this adds a considerable number of files to the Winchester disk, it is also advisable for you to take advantage of the hierarchical structure of iRMX 86 files and create some new directories to contain these files.

First, create under the root directory a directory called RMX to contain all your configuration files. Next, create additional directories under RMX to contain the files for each iRMX 86 layer. Name these directories NUCLEUS, TH, BIOS, EIOS, LOADER, and HI. Finally, copy the files from each release diskette to files of the same name in the corresponding directory. For example, copy all files on the Terminal Handler and Debugger Release Diskette to files in the directory RMX/TH. Copy all files on the Human Interface Release Diskette to files in the directory RMX/HI. Continue this process for the remaining diskettes.

By copying the files from the release diskettes and maintaining them in the logical structure just outlined, you may find you can more easily locate these files when you need them. Your directory displays will also be shorter and less cluttered.

MODIFYING THE FILES FOR SYSTEM 86/330 COMPATIBILITY

Each iRMX 86 release diskette contains one or more configuration files (with names ending in ".P86" or ".A86") and one or more SUBMIT files (with names ending in ".CSD"). These files contain $INCLUDE statements or commands that make direct reference to other files. These statements and commands use the terms :F0:, :F1:, :F2:, and :F3: to refer to the drives on which the diskettes are mounted. The iRMX 86 CONFIGURATION GUIDE discusses the drive conventions assumed by the release diskettes.

Although the drive conventions are useful when performing the configuration process on a system with four flexible disk drives, they do not permit you to perform the configuration process on the System 86/330 Microcomputer System. Thus, before you can configure your customized version of the Operating System, you must make modifications to the configuration and SUBMIT files.

To make the necessary changes, examine all files whose names end with the characters ".P86", ".A86", and ".CSD". In each of these files, look for references to :F0:, :F1:, :F2:, and :F3:. You must change each of these terms to the appropriate combination of logical name and directory names to reflect the actual pathname of the file.

For example, as released, one of the Extended I/O System configuration files, ETABLE.A86, contains the statement:

```
$INCLUDE(:F2:ETABLE.MAC)
```

If you have copied the release diskettes to the Winchester disk as recommended in the last section, you must modify this statement to read:

```
$INCLUDE(:WD0:RMX/EIOS/ETABLE.MAC)
```

You must make similar changes for all references to :F0:, :F1:, :F2:, and :F3:.

PRECONFIGURED VERSION OF THE OPERATING SYSTEM

This section describes the changes that you must make to the iRMX 86 configuration files in order to produce an application system that is exactly like the preconfigured version of the Operating System. With this information, you can include some or all of the features of the preconfigured system in your customized system.

This section is divided into several subsections: one for each layer of the Operating System that requires configuration changes, one for the System Debug Monitor, and one for the root job. You should read these subsections in conjunction with the iRMX 86 CONFIGURATION GUIDE.

Nucleus Configuration

Two configuration files exist for the Nucleus: NTABLE.A86 and NDEVCF.A86. To create the preconfigured version of the Operating System, you need to change only NDEVCF.A86. For this file, you must include the %NDP_SUPPORT macro, as follows:

```
%NDP_SUPPORT(08h)
```

Basic I/O System Configuration

Two configuration files exist for the Basic I/O System: ITABLE.A86 and IDEVCF.A86. To create the preconfigured version of the Operating System, you need to change only IDEVCF.A86. For this file, you must make the following changes:

- Delete the DUIBs for devices which you do not intend to include as part of your System 86/330 hardware. In the preconfigured version of the Operating System, all DUIBs have been deleted from IDEVCF.A86 except those with the following names:

TO	BB
IWO	STREAM
WFO	WF1
WFDO	WFD1
WFDDO	WFDD1

  The software will function correctly even if you do not delete the other DUIBs; however, to conserve memory, you should delete DUIBs for devices that you have no intention of using.

- Add DUIBs for devices which are not included in IDEVCF.A86. The preconfigured version of the Operating System adds one DUIB to support a second Winchester disk. To include this support in your application system, add the following information to IDEVCF.A86:

```
define_duib <
&     'IW1',            ; Name(14)
&     00Bh,             ; file$drivers
&     0FFh,             ; functs
&     00h,              ; flags
&     1024,             ; dev$gran  Sector Size in bytes
&     02000h, 001DEh,   ; dev$size = 1024*12*5*510
&     1,                ; device
&     1,                ; unit
&     7,                ; dev$unit
&     initio,           ; init$io
&     finishio,         ; finish$io
&     queueio,          ; queue$io
&     cancelio,         ; cancel$io
&     dinfo_215,        ; device$info
&     uinfo_215iw,      ; unit$info
&     050,              ; update$timeout
&     6,                ; num$buffers
&     129               ; priority
&>
```

- Adjust the device_unit_number parameters of all the DUIB structures in IDEVCF.A86 to account for the DUIBs that you added and removed. Device-unit numbers must be sequential and start with 0. All unique units must have unique device-unit numbers. If there are multiple DUIBS for one device-unit, each DUIB must have the same device-unit number.

- For each DUIB in IDEVCF.A86, adjust the values of the following parameters:

Parameter	New value
update_timeout	50
num_buffers	6

Extended I/O System Configuration

Three configuration files exist for the Extended I/O System: ETABLE.A86, EDEVCF.A86, and EJOBCF.A86. To create the preconfigured version of the Operating System, you need to change only EDEVCF.A86. You must modify this file to contain the following information:

```
;     BYTE-BUCKET
;
          %DEV_INFO_BLOCK('BB', 'BB', PHYSICAL)
;
;     TERMINAL
;
          %DEV_INFO_BLOCK('TO', 'TO', PHYSICAL)
;
;     215 WINCHESTER, UNIT 0, DRIVE 0
;
          %DEV_INFO_BLOCK('WD0', 'IW0', NAMED)
;
;     218 WINCHESTER FLOPPY DS/DD, UNIT 0, DRIVE 0
;
          %DEV_INFO_BLOCK('FD0', 'WFDD0', NAMED)
;
;     STREAM
;
          %DEV_INFO_BLOCK('STREAM', 'STREAM', STREAM)
;
;     LP
;
          %DEV_INFO_BLOCK('LP', 'LP', PHYSICAL)

          %END_DEV_CONFIG(1024)

               END
```

Application Loader Configuration

The only configuration file for the Application Loader is LCONFG.P86. You must modify one line of this file to specify a larger default memory pool size. The affected line after modification should appear as:

```
DECLARE L$DEFAULT$MEMPOOL  WORD PUBLIC DATA(200H);   /*PAGES */
```

Human Interface Configuration

The Human Interface configuration file is named HCONFG.P86. You must modify this file if you want provide the features of the preconfigured version of the Operating System. However, the modifications differ depending on whether you want your system disk to be the Winchester disk or the flexible disk. If you want the Winchester disk to be your system disk, you should modify HCONFG.A86 to include the following lines:

```
DECLARE
     H$system$directory(*)     BYTE PUBLIC DATA(11, ':WD0:SYSTEM'),
     H$prog$directory(*)       BYTE PUBLIC DATA(14, ':WD0:USER/PROG'),
     H$default$dir(*)          BYTE PUBLIC DATA(09, ':WD0:USER'),
     H$work$directory(*)       BYTE PUBLIC DATA(14, ':WD0:USER/WORK');

DECLARE
     H$sign$on(*)   BYTE PUBLIC DATA(18, 'SYSTEM 86/330 V1.0'),

     H$command$name$max  WORD PUBLIC DATA(50),

     H$prefixes(*)  BYTE PUBLIC DATA(4,  /*number of prefixes */
          03,  ':$:',          /* first prefix */
          06,  ':PROG:',       /* second prefix */
          08,  ':SYSTEM:',     /* third prefix */
          10,  ':WD0:LANG/';   /* fourth prefix */
```

If you want your system disk to be the flexible disk, change all occurrences of :WD0: in the previous lines to :FD0:.

Regardless of the disk you choose for your system disk, you must also modify one additional line of HCONFG.P86 to specify the LOGIN file. Although this line appears in the portion of the file which is reserved for fixed information, you should change the declaration of the variable H$z$command to:

```
H$z$command(*)  BYTE PUBLIC DATA(12, 'SUBMIT LOGIN');
```

System Debug Monitor Configuration

The preconfigured version of the Operating System includes the System Debug Monitor (SDB) as a first-level job (refer to the iRMX 86 CONFIGURATION GUIDE for information about first-level jobs). The SDB has one configuration file called SDBCNF.A86, which you can modify to set the interrupt level. To be compatible with the preconfigured version of the Operating System, set the value of the variable "level" as follows:

```
level  dw  018h
```

System Configuration File

After configuring each of the Operating System layers and the SDB, you must create a system configuration file for use in configuring the root job. To create the preconfigured version of the Operating System, this file should contain the following information (assuming you set up your directory structure as described previously):

```
        name sroot

    $include(RMX/NUCLEUS/ctable.mac)

    ;  SDB job:

    %job(0,                     %' object directory size
         0100h, 0100h,          %' pool size (min., max.)
         010h, 010h,            %' max objects and tasks
         0,                     %' max job priority
         0:0, 0,                %' exception handler addr, mode
         0,                     %' job flags
         130,                   %' init task priority
         1D0D:0,                %' init task entry address
         0,                     %' init task data segment address
         0:0, 100h,                 %' init task stack address, stack size
         0)                     %' init task flags

    ; IOS job:

    %job(0,                     %' object directory size
         0600h, 0820h,          %' pool size (min., max.)
         0FFFFh, 0FFFFh,        %' max objects and tasks
         0,                     %' max job priority
         0:0, 0,                %' exception handler addr, mode
         0,                     %' job flags
         130,                   %' init task priority
         719:0,                 %' init task entry address
         0,                     %' init task data segment address
         0:0, 200h,                 %' init task stack address, stack size
         0)                     %' init task flags

    ;  LOADER JOB:

    %job(0,                     %' object directory size
         020h, 020h,            %' pool size (min., max.)
         50, 5,                 %' max objects and tasks
         0,                     %' max job priority
         0:0, 0,                %' exception handler addr, mode
         0,                     %' job flags
         130,                   %' init task priority
         16A4:0,                %' init task entry address
         0,                     %' init task data segment address
         0:0, 160,              %' init task stack address, stack size
         0)                     %' init task flags
```

```
; EIOS job:

%job(0,                         %' object directory size
     0150h, 0FFFFh,             %' pool size (min., max.)
     0FFFFh, 0FFFFh,            %' max objects and tasks
     130,                       %' max job priority
     0:0, 0,                    %' exception handler addr, mode
     0,                         %' job flags
     150,                       %' init task priority
     13B9:0,                    %' init task entry address
     0,                         %' init task data segment address
     0:0, 250h,                 %' init task stack address, stack size
     0)                         %' init task flags

;    remove memory for:
;         interrupt vectors (0 - 3FFh)
;         iSBC 957B data area (400h - 7FFh)
;         SDB data area (800h - 0CCFh)
;    NOTE: free space 1 (0CD0h - 0FFFFh)
;    remove memory for:
;         iRMX 86 Operating System (1040h - 21C4Fh)
;              Nucleus
;              Basic I/O System
;              Application Loader
;              Extended I/O System
;              Human Interface
;              root job
;              SDB
;    NOTE: free space 2 (21C50h - 0FBFFFh)
;    remove memory for:
;         SCT PROM address (0FC000h - 0FFFFFh)

%sab(0h, 0CDh, U)
%sab(100h, 103h, U)
%sab(104h, 21C5h, U)
%sab(0FC00h, 0FFFFh, U)

%system(104, 30, 40h, A, N, 3)

     END
```

INDEX

Underscored entries are primary references.

intel®

REQUEST FOR READER'S COMMENTS

Intel Corporation attempts to provide documents that meet the needs of all Intel product users. This form lets you participate directly in the documentation process.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this document.

1. Please specify by page any errors you found in this manual.

2. Does the document cover the information you expected or required? Please make suggestions for improvement.

3. Is this the right type of document for your needs? Is it at the right level? What other types of documents are needed?

4. Did you have any difficulty understanding descriptions or wording? Where?

5. Please rate this document on a scale of 1 to 10 with 10 being the best rating. _____

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

CITY _____ STATE _____ ZIP CODE _____

Please check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS . . .

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 79 BEAVERTON, OR

POSTAGE WILL BE PAID BY ADDRESSEE

Intel Corporation
5200 N.E. Elam Young Pkwy.
Hillsboro, Oregon 97123

O.M.S. Technical Publications

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) 987-8080

Printed in U.S.A.